Type-Based Complexity Analysis of Probabilistic Functional Programs (Technical Report)

Martin Avanzini, Ugo Dal Lago, Alexis Ghyselen

Type-Based Complexity Analysis of Probabilistic Functional Programs (Technical Report)

MARTIN AVANZINI, INRIA Sophia Antipolis
UGO DAL LAGO, University of Bologna, INRIA Sophia Antipolis
ALEXIS GHYSELEN, ENS Lyon

We show that complexity analysis of probabilistic higher-order functional programs can be carried out compositionally by way of a type system. The introduced type system is a significant extension of linear dependent types. On the one hand, the presence of probabilistic effects requires adopting a form of *dynamic* distribution type, subject to a coupling-based subtyping discipline. On the other hand, recursive definitions are proved terminating by way of ranking functions. We prove not only that the obtained system, called dℓRPCF, provides a sound methodology for average case complexity analysis, but is also *extensionally complete*, in the sense that all average case polytime Turing machines can be encoded as a term typable in dℓRPCF.

1 A PROBABILISTIC FUNCTIONAL LANGUAGE EXTENDING AFFINE PCF

In this section we introduce the programming language dℓRPCF that we consider throughout this work. It is basically an affine version of Plotkin's, extended with an operator `Unif` for sampling from uniform, discrete distributions. The type system we introduce in this section does not guarantee any complexity propriety.

Statics: The sets of *terms*, *values* and *types* are generated by the following grammars:

$$\begin{array}{lll} \text{Terms} & t, u ::= v \mid v\ w \mid \mathtt{let}\ x = t\ \mathtt{in}\ u \mid \mathtt{match}\ v\ \mathtt{with}\ \{\ \mathsf{z} \mapsto\ t \mid \mathsf{s} \mapsto\ w\ \} \\ & \qquad \mid \mathtt{let}\ \langle x, y\rangle = v\ \mathtt{in}\ t \\ \text{Values} & v, w ::= x \mid \mathsf{z} \mid \mathsf{s}(v) \mid \mathtt{Unif} \mid \lambda x.t \mid \mathtt{fix}\ x.v \mid \langle v, w\rangle \\ \text{Types} & T, U ::= \mathsf{Nat} \mid T \multimap U \mid T \otimes U \end{array}$$

Terms are restricted to A-normal forms. In this setting, the application $t\ u$ is recovered by $\mathtt{let}\ x = t\ \mathtt{in}\ \mathtt{let}\ y = u\ \mathtt{in}\ x\ y$. Apart from our sampling operator `Unif`, the constructors are standard. We follow the usual convention where application $t\ u$ binds to the left, whereas λ-abstraction $\lambda.t$ binds to the right. Terms which are not values are called *active*. For an integer $n \in \mathbb{N}$, we denote with $\underline{n}$ the value $\mathsf{s}(\ldots, \mathsf{s}(\mathsf{z}), \ldots)$, with n occurrences of s. The capture free substitution of variable x by value v in t is denoted $t[x := v]$.

We impose a linear typing regime on terms. The typing rules are presented in Figure 1. The statement $\Gamma \mid \Theta \vdash t : T$ means that under the *(linear) typing context* Γ and the *global typing context* Θ the term t receives the type T. Here, a typing context Γ is a non-ordered sequence of the form $x_1 : T_1 \ldots x_n : T_n$. The union of two linear type contexts Γ and Δ, denoted Γ, Δ, is defined only if for each variable x with $(x : T) \in \Gamma$ and $(x : U) \in \Delta$, it holds that $T = U = \mathsf{Nat}$. Global type contexts Θ, used to treat recursive functions, are either empty or consist of a unique hypothesis $x : T \multimap U$. In rule Fix, $\ell\Gamma$ denotes a typing context where all variables are given the type Nat. For closed terms t, we abbreviate $\varnothing \mid \varnothing \vdash t : T$ by $\vdash t : T$. We remark that this affine type system permits duplication of values of base type, whereas duplication of values of functional types is prohibited.

Authors' addresses: Martin Avanzini, INRIA Sophia Antipolis; Ugo Dal Lago, University of Bologna, INRIA Sophia Antipolis; Alexis Ghyselen, ENS Lyon.

2019. XXXX-XXXX/2019/1-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

$$\frac{(x:T)\in\Gamma \text{ or } (x:T)\in\Theta}{\Gamma \mid \Theta \vdash x : T}\,(\textsc{Ax}) \qquad \frac{}{\Gamma \mid \Theta \vdash \mathsf{z} : \mathsf{Nat}}\,(\textsc{Zero}) \qquad \frac{\Gamma \mid \Theta \vdash v : \mathsf{Nat}}{\Gamma \mid \Theta \vdash \mathsf{s}(v) : \mathsf{Nat}}\,(\textsc{Succ})$$

$$\frac{}{\Gamma \mid \Theta \vdash \mathtt{Unif} : \mathsf{Nat} \multimap \mathsf{Nat}}\,(\textsc{Unif}) \qquad \frac{\Gamma \mid \Theta \vdash v : T \multimap U \quad \Delta \mid \Theta \vdash w : T}{\Gamma,\Delta \mid \Theta \vdash v\ w : U}\,(\textsc{App})$$

$$\frac{\Gamma, x:T \mid \Theta \vdash t : U}{\Gamma \mid \Theta \vdash \lambda x.t : T \multimap U}\,(\textsc{Abs}) \qquad \frac{\ell\Gamma \mid (x : T \multimap U) \vdash v : T \multimap U}{\Gamma, \ell\Gamma \mid \Theta \vdash \mathtt{fix}\ x.v : T \multimap U}\,(\textsc{Fix})$$

$$\frac{\Gamma \mid \Theta \vdash t : T \quad \Delta, x:T \mid \Theta \vdash u : U}{\Gamma,\Delta \mid \Theta \vdash \mathtt{let}\ x = t\ \mathtt{in}\ u : U}\,(\textsc{Let}) \qquad \frac{\Gamma \mid \Theta \vdash v : T \quad \Delta \mid \Theta \vdash w : U}{\Gamma,\Delta \mid \Theta \vdash \langle v, w\rangle : T \otimes U}\,(\otimes_i)$$

$$\frac{\Gamma \mid \Theta \vdash v : T \otimes T' \quad \Delta, x:T, y:T' \mid \Theta \vdash t : U}{\Gamma,\Delta \mid \Theta \vdash \mathtt{let}\ \langle x, y\rangle = v\ \mathtt{in}\ t : U}\,(\otimes_e)$$

$$\frac{\Gamma \mid \Theta \vdash v : \mathsf{Nat} \quad \Delta \mid \Theta \vdash t : T \quad \Delta \mid \Theta \vdash w : \mathsf{Nat} \multimap T}{\Gamma,\Delta \mid \Theta \vdash \mathtt{match}\ v\ \mathtt{with}\ \{\ \mathsf{z} \mapsto\ t \mid \mathsf{s} \mapsto\ w\ \} : T}\,(\textsc{Match})$$

Fig. 1. Affine type system for probabilistic PCF

Dynamics: Following [Dal Lago and Grellois 2017], we can give dℓRPCF an operational semantics in terms of a binary relation $\Rightarrow$ on distributions. On the non-probabilistic fragment of our language, i.e., on terms without any occurrences of `Unif`, the semantics can be seen isomorphic to the usual (weak) call-by-value reduction relation.

A *(discrete) valuation* on a countable set X is a function $v : X \to [0, +\infty]$. The *support* of v is given by $Supp(v) \triangleq \{x \in X \mid v(x) > 0\}$. It is called *finite* if its support is finite. The set of valuations on X is denoted $V(X)$. We may denote a valuation v also by $\{v(x) : x\}_{x \in Supp(v)}$ or $\{v(x_1) : x_1, \ldots, v(x_n) : x_n\}$ when $Supp(v) = \{x_1, \ldots, x_n\}$ is finite. Scalar multiplication $p \cdot v$ and finite sum $\sum_{i\in I} v_i$ are defined point-wise. A *(discrete) distribution* on X is a valuation $\mathcal{D} : X \to [0, 1]$ such that $\sum \mathcal{D} \triangleq \sum_{x\in X} \mathcal{D}(x) \leq 1$. It is called *proper* if $\sum \mathcal{D} = 1$. The set of distributions is closed under scalar multiplication, but not necessarily under finite sum. We define the relation $\leq$ on distributions such that $\mathcal{D} \leq \mathcal{E}$ if $\mathcal{D}(x) \leq \mathcal{E}(x)$ for all $x \in X$. For a distribution $\mathcal{D}$ on terms, $\mathcal{D}_a +_{a/v} \mathcal{D}_v$ indicates the decomposition of $\mathcal{D}$, i.e. $\mathcal{D} = \mathcal{D}_a + \mathcal{D}_v$, so that the supports of $\mathcal{D}_a$ and $\mathcal{D}_v$ consist of active terms and values, respectively.

The reduction relation $\Rightarrow$ is itself based on an auxiliary relation $\to$, depicted in Figure 2, which maps active terms to distributions. If $t \to \{p_1 : t_1, \ldots, p_n : t_n\}$, then t_i should be understood as a one-step reduct of t with probability p_i. All rules but the one defining `Unif` follow the standard operational semantics of PCF. A term `Unif` $\underline{n}$ reduces to the uniform distribution on $(0, n)$ wrt. $\to$. Based on the auxiliary relation $\to$, the relation $\Rightarrow$ is given by the following inference rule.

$$\frac{\mathcal{D} = \{p_i : t_i \mid i \in \mathcal{I}\} +_{a/v} \mathcal{D}_v \quad \forall i \in \mathcal{I}, t_i \to \mathcal{E}_i}{\mathcal{D} \Rightarrow \mathcal{D}_v + \sum_{i\in\mathcal{I}} p_i \cdot \mathcal{E}_i}$$

The type system is coherent with the reduction rules, in particular, the type system enjoys subject reduction and progress in the following sense.

Proposition 1.1 (Subject Reduction and Progress). *Let t be such that $\vdash t : T$ holds.*

(1) *If $t \to \{p_i : t_i \mid i \in \mathcal{I}\}$ (with $p_i > 0$ for all i) then for all $i \in \mathcal{I}$, we have $\vdash t_i : T$.*
(2) *The term t is in normal form for $\to$ if and only if t is a value.*

$$\frac{}{(\lambda x.t)\ v \to \{1 : t[x := v]\}} \qquad \frac{}{(\mathtt{fix}\ x.w)\ v \to \{1 : w[x := \mathtt{fix}\ x.w]\ v\}}$$

$$\frac{}{\mathtt{Unif}\ \underline{n} \to \{\tfrac{1}{n+1} : \underline{m} \mid 0 \le m \le n\}}$$

$$\frac{}{\mathtt{match\ z\ with}\ \{\ \mathsf{z} \mapsto\ t \mid \mathsf{s} \mapsto\ w\ \} \to \{1 : t\}} \qquad \frac{}{\mathtt{match\ s}(v)\ \mathtt{with}\ \{\ \mathsf{z} \mapsto\ t \mid \mathsf{s} \mapsto\ w\ \} \to \{1 : w\ v\}}$$

$$\frac{}{\mathtt{let}\ x = v\ \mathtt{in}\ t \to \{1 : t[x := v]\}} \qquad \frac{}{\mathtt{let}\ \langle x, y\rangle = \langle v, w\rangle\ \mathtt{in}\ t \to \{1 : t[x := v][y := w]\}}$$

$$\frac{t \to \{p_i : t_i \mid i \in \mathcal{I}\}}{\mathtt{let}\ x = t\ \mathtt{in}\ u \to \{p_i : \mathtt{let}\ x = t_i\ \mathtt{in}\ u \mid i \in \mathcal{I}\}}$$

Fig. 2. Reductions rules on distributions

Subject reduction, i.e. Property 1.1.1, implies in particular that distributions of well typed terms are closed under $\Rightarrow$ reductions.

We denote with $\Rightarrow^*$ the reflexive and transitive closure of $\Rightarrow$, and we denote with $\Rightarrow^n$ the $n^t h$ iteration of the reduction relation $\Rightarrow$. We remark that the set of finite distributions is closed under $\Rightarrow$. Finally, let $\mathcal{D} \Rightarrow_v \mathcal{E}_v$ if $\mathcal{D} \Rightarrow \mathcal{E}_a +_{a/v} \mathcal{E}_v$, and similarly the relations $\Rightarrow_v^n$ and $\Rightarrow_v^*$ are defined in terms of $\Rightarrow^n$ and $\Rightarrow^*$, respectively. Note that if $\mathcal{D} \Rightarrow_v^n \mathcal{E}_n$ where $n \in \mathbb{N}$, then $\mathcal{E}_m \le \mathcal{E}_n$ whenever $m \le n$. Likewise, we define the relation $\Rightarrow_a$. The *semantics* of a term is a distribution on values defined as $[\![t]\!] = \sup\{\mathcal{D} \mid t \Rightarrow_v^* \mathcal{D}\}$. This is a well-posed definition because distributions form an ω**CPO**.

Example 1.2 (Biased Random Walk). For two terms t and u, let us denote by $t \oplus_n u$ the term $\mathtt{let}\ p = \mathtt{Unif}\ \underline{n}\ \mathtt{in}\ (\mathtt{match}\ p\ \mathtt{with}\ \{\ \mathsf{z} \mapsto\ t \mid \mathsf{s} \mapsto\ \lambda q.u\ \})$ for fresh variables p and q. Then $t \oplus_n u$ reduces with probability $\frac{1}{n+1}$ to t, and with probability $\frac{n}{n+1}$ to u. Consider the term

$$\mathtt{rwalk} \triangleq \mathtt{fix}\ rw.\lambda n.\mathtt{match}\ n\ \mathtt{with}\ \{\ \mathsf{z} \mapsto\ \mathsf{z} \mid \mathsf{s} \mapsto\ \lambda m.rw\ (\mathsf{s}(\mathsf{s}(m))) \oplus_2 rw\ m\ \}\ .$$

Its recursive calls gives a biased random walk, with $\mathtt{rwalk}\ \underline{n+1}$ reducing to $\mathtt{rwalk}\ \underline{n+2}$ with probability $\frac{1}{3}$, and with probability $\frac{2}{3}$ to $\mathtt{rwalk}\ \underline{n}$. For any $n \in \mathbb{N}$, $\mathtt{rwalk}\ \underline{n}$ reduces to z almost surely, and hence $[\![\mathtt{rwalk}\ \underline{n}]\!] = \{1 : \mathsf{z}\}$.

In this work, we are interested in average case complexity analysis, in terms of reduction steps. In a probabilistic setting, the reduction length from a term t can be understood as a random variable $\mathcal{S}_t$ on $\mathbb{N} \cup \{\infty\}$, with $\mathbb{P}(\mathcal{S} = n)$ being the probability that t evaluates to normal form in n steps, or diverges in the case $n = \infty$. The expected runtime of a term t is then defined in terms of its expectation

$$\mathsf{E}(\mathcal{S}_t) \triangleq \sum_{n=0}^{\infty} \mathbb{P}(\mathcal{S}_t > n)\ .$$

Here, $\mathbb{P}(\mathcal{S}_t > n)$ gives the probability that a reduction takes strictly more than n steps. In our setting, this probability is expressed by $\sum \mathcal{D}_a^n$, for $\mathcal{D}_a^n$ the distribution of active terms reachable in n steps from t, i.e., $t \Rightarrow_a^n \mathcal{D}_a^n$. This motivates the following definition, compare [Avanzini et al. 2018] for further justification of this definition.

Definition 1.3 (Expected Runtime). The *expected runtime* of a term t is defined by $\mathbb{E}(t) \triangleq \sum_{n=1}^{\infty}(\sum \mathcal{D}_a^n)$ where $\mathcal{D}_a^n$ is the distribution on active terms such that $t \Rightarrow_a^n \mathcal{D}_a^n$.

We remark that $\mathbb{E}(v) = 0$ for any value v, and if $t \to \{p_i : t_i \mid i \in \mathcal{I}\}$ then $\mathbb{E}(t) = 1 + \sum_{i\in\mathcal{I}} p_i \cdot \mathbb{E}(t_i)$.

2 THE TYPE SYSTEM

This section is devoted to introducing the main object of study of this paper, dℓRPCF, a monadic, linear dependent type system for reasoning about expected runtimes. This system borrows ideas from the dependent type system dℓPCF introduced by Dal Lago and Petit [Dal Lago and Petit 2012] for reasoning about the runtime of deterministic programs and from the affine, monadic type system introduced by Dal Lago and Grellois [Dal Lago and Grellois 2017] for proving almost sure termination of probabilistic programs.

2.1 Indexes, Types and Subtyping

As in the case of linear dependent types, base types are annotated with refinement constraints. Constraints are formed over *index terms*, i.e., first-order terms generated freely from a set of *index symbols* $\mathcal{I}$ and *index variables* $\mathcal{V}$, denoted by $f, g, \dots$ and $a, b, \dots$ respectively. Each symbol $f \in \mathcal{I}$ is associated with a natural number $\mathrm{ar}(f)$, its *arity*.

Definition 2.1 (Indices). Natural indices $I, J, \dots$ and rational indices $P, Q, \dots$ are generated from function symbols $\mathcal{I}$ and index variables $\mathcal{V}$ according to the following grammar:

$$I, J \triangleq a \mid f(I_1, \dots, I_{\mathrm{ar}(f)}) \mid \sum_{a\le I} J \mid max_{a\le I} J \qquad\qquad P, Q \triangleq \frac{I}{J}\,.$$

We use $A, B, \dots$ to denote natural and rational indices. We assume that each function symbol $f \in \mathcal{I}$ comes equipped with an interpretation $[\![f]\!] : \mathbb{N}^{\mathrm{ar}(f)} \to \mathbb{N}$. We do not put any constraints on these functions, apart from computability. Given a valuation $\rho : \mathcal{V} \to \mathbb{N}$, the interpretation of symbols is extended homomorphically to indices: $[\![a]\!]_\rho \triangleq \rho(a)$ for index variables a, $[\![f(I_1, \dots, I_n)]\!]_\rho \triangleq [\![f]\!]([\![I_1]\!]_\rho, \dots, [\![I_n]\!]_\rho)$ and $[\![\frac{I}{J}]\!]_\rho \triangleq \frac{[\![I]\!]_\rho}{[\![J]\!]_\rho}$ if $[\![J]\!]_\rho \neq 0$. In the case $[\![J]\!]_\rho = 0$, $[\![\frac{I}{J}]\!]_\rho$ is undefined. Note that for any valuation ρ and index A, $[\![A]\!]_\rho$ is computable. We suppose that $\mathcal{I}$ contains for each $n \in \mathbb{N}$ a constant n interpreted by $[\![\mathsf{n}]\!] \triangleq n$ as well as symbols $+$ and $\cdot$ interpreted as addition and multiplication, respectively. For an index I over variables $a_1, \dots, a_k$ and J over $a_1, \dots, a_k, b$, we define a special symbol $\sum_{b\le I} J$ with an interpretation satisfying $[\![\sum_{b\le I} J]\!]_\rho = \sum_{n\le [\![I]\!]_\rho} [\![J]\!]_{\rho[b\mapsto n]}$. Likewise, we define the maximum of a bounded sequence. We will often, by an abuse of notation, use operations on rational indexes. For example, $\frac{I}{J} + \frac{I'}{J'}$ denotes the rational index $\frac{I\cdot J' + I'\cdot J}{J\cdot J'}$. For an index A, we define the substitution of a in A by a natural index J, that we note $A\{J/a\}$, in the obvious way.

Definition 2.2 (Constraints on Indexes). Let $\phi \subseteq \mathcal{V}$ be a set of index variables. A *constraint* C on ϕ is an expression of the form $A\ \mathcal{R}\ B$ where A, B are indexes with free variables in ϕ. Here, $\mathcal{R}$ denotes a relation between integers or rationals. Usually, we use relations in the set $\{\le, <, =, \neq\}$ but we could use any computable relation. Finite sets of constraints are denoted by Φ. We say that an index valuation $\rho : \phi \to \mathbb{N}$ satisfies a constraint $A\ \mathcal{R}\ B$, in notation $\rho \vDash A\ \mathcal{R}\ B$, if $[\![A]\!]_\rho$ and $[\![B]\!]_\rho$ are defined and $[\![A]\!]_\rho\ \mathcal{R}\ [\![B]\!]_\rho$ holds. Likewise, $\rho \vDash \Phi$ if $\rho \vDash A\ \mathcal{R}\ B$ holds for all $(A\ \mathcal{R}\ B) \in \Phi$. In the same way, we say that $\phi; \Phi \vDash A\ \mathcal{R}\ B$ when for any $\rho : \phi \to \mathbb{N}$ such that $\rho \vDash \Phi$ then $\rho \vDash A\ \mathcal{R}\ B$.

Definition 2.3 (Linear Dependent Types). Linear dependent types σ, τ and *dynamic distribution types* (DDTs) μ, ν are defined as follows:

$$\begin{aligned} \text{linear dependent types} \qquad & \sigma, \tau \triangleq \mathsf{Nat}(a \mid \Phi) \mid \sigma \otimes \tau \mid \sigma_{\multimap} \\ \text{arrow types} \qquad & \sigma_{\multimap} \triangleq \sigma \multimap \mu \mid \forall a : \Phi.\sigma_{\multimap} \\ \text{dynamic distribution types} \qquad & \mu, \nu \triangleq \{P : \sigma \mid a \leq I\} \end{aligned}$$

A type $\mathsf{Nat}(a \mid \Phi)$ represents the set of naturals n for which $\Phi\{\mathsf{n}/a\}$ is true. It should thus be understood as an existential type binding a, with a occurring free in Φ. For instance, $\mathsf{Nat}(a \mid a \leq \mathsf{n})$ represents a natural number between zero and $n \in \mathbb{N}$, and more generally, $\mathsf{Nat}(a \mid a \leq J)$ would represent naturals bounded by J. For brevity, we may abbreviate with $\mathsf{Nat}(I)$ the type $\mathsf{Nat}(a \mid a = I)$, specifically, $\mathsf{Nat}(b)$ denotes $\mathsf{Nat}(a \mid a = b)$.

Our type system admits polymorphism over indices in the form of bounded universal quantification over function types. The variable a in a type $\forall a : \Phi.\sigma_{\multimap}$ can be free in Φ and $\sigma_{\multimap}$, whereas it is bound in $\forall a : \Phi.\sigma_{\multimap}$. We will sometimes abbreviate an arrow type $\sigma_{\multimap}$ as $\forall \overline{a} : \overline{\Phi}.\sigma \multimap \mu$, with $\overline{a}$ being a list of index variables and $\overline{\Phi}$ a list of sets of constraints.

Finally, the dynamic distribution type $\{P : \sigma \mid a \leq I\}$ can be understood as a monadic type for probabilistic computations which yield with probability P an element of type σ.

In the dynamic distribution type, a can be free in P and σ but not in I, and it will be considered bound in $\{P : \sigma \mid a \leq I\}$. For instance, the DDT $\{\frac{1}{I+1} : \mathsf{Nat}(b) \mid b \leq I\}$ represents a probabilistic computation that evaluates to a natural number uniformly distributed in the interval from 0 to I. This is indeed the type that our system will assign to the term `Unif` t, where t is of type $\mathsf{Nat}(I)$. We may abbreviate with σ the DDT $\{1 : \sigma \mid a \leq 0\}$ representing a dirac distribution, for a not occurring free in σ. Thereby, we may assign to terms of function type that exhibit no probabilistic behaviour the usual type $\tau \multimap \sigma$, instead of $\tau \multimap \{1 : \sigma \mid a \leq 0\}$.

Types in general are indicated by $\zeta, \xi, \ldots$. We consider types equal modulo renaming of bound variables and denote by $\zeta\{I/a\}$ the capture-avoiding substitution of the index variable a by I in the type ζ. All types are defined under some restrictions, such as the fact that the sum of probabilities in a distribution must be equal to 1. These restrictions are captured in our notion of valid type, see Figure 3. Notice that if an index A is valid under $\phi; \Phi$, then for all valuations $\rho : \phi \to \mathbb{N}$ such that $\rho \vDash \Phi$, $[\![A]\!]_\rho$ is well defined. From now on, given a context $\phi; \Phi$, we will only consider valid types without always reminding it.

2.2 Subtyping

Since our type annotations give a form of refinement, it should always be possible to relax a refinement to a more liberal one. To this end, we introduce a subtyping relation on types. The core of our subtyping relation, presented in Figure 4 is fairly standard. A type $\mathsf{Nat}(a \mid \Phi_1)$ is a subtype of $\mathsf{Nat}(a \mid \Phi_2)$ if Φ_1 implies Φ_2. One natural way to extend subtyping to DDTs is to lift the subtyping relation component-wise, i.e. $\{P : \sigma \mid a \leq I\}$ is a subtype of $\{P : \tau \mid a \leq I\}$ if σ is a subtype of τ, for all $a = 0, \ldots, I$. Although simple, this extension is too rigid to deal with more interesting functions. Instead, our treatment of subtyping for DDTs is based on the notion of *probabilistic coupling*, already studied in computer science [Barthe et al. 2017].

Intuitively, in the coupling $S \triangleleft_{\sqsubseteq} \langle \{P : \sigma \mid a \leq I\} \& \{Q : \tau \mid b \leq J\} \rangle$, S can be seen as a distribution over the set $\{(\sigma, \tau) \mid a \leq I, b \leq J, \sigma \mathcal{R} \tau\}$ and S denotes the fraction of the probability P for σ that will contribute to the probability Q for τ. For example, if $\sigma \mathcal{R} \tau$ and $\sigma' \mathcal{R} \tau$, then $\{\frac{1}{2} : \sigma, \frac{1}{2} : \sigma'\}$ and $\{\frac{1}{4} : \sigma, \frac{1}{4} : \sigma', \frac{1}{2} : \tau\}$ are coupled by the distribution $\{\frac{1}{4} : (\sigma, \tau), \frac{1}{4} : (\sigma', \tau), \frac{1}{4} : (\sigma, \sigma), \frac{1}{4} : (\sigma', \sigma')\}$.

Subtyping is extended to function types in the standard way: $\sigma \multimap \mu \sqsubseteq \tau \multimap \nu$ holds if $\mu \sqsubseteq \nu$ and $\sigma \sqsubseteq \tau$. Then for quantification, on the right we internalize quantification in the set of variables

Indices and constraints:

$$\frac{\mathsf{FV}(I) \subseteq \phi}{\phi;\Phi \vdash I \ valid} \qquad \frac{\phi;\Phi \vdash I \ valid \quad \phi;\Phi \vdash J \ valid \quad \phi;\Phi \vDash J \neq 0}{\phi;\Phi \vdash \frac{I}{J} \ valid}$$

$$\frac{\phi;\Phi \vdash A \ valid \quad \phi;\Phi \vdash B \ valid}{\phi;\Phi \vdash A \mathcal{R} B \ valid} \qquad \frac{\phi;\Phi \vdash A \mathcal{R} B \ valid \text{ for all } (A \mathcal{R} B) \in \Phi'}{\phi;\Phi \vdash \Phi' \ valid}$$

Types:

$$\frac{a \notin \phi \quad (\phi, a);\Phi \vdash \Phi_a \ valid}{\phi;\Phi \vdash \mathsf{Nat}(a \mid \Phi_a) \ valid} \qquad \frac{\phi;\Phi \vdash \sigma \ valid \quad \phi;\Phi \vdash \tau \ valid}{\phi;\Phi \vdash \sigma \otimes \tau \ valid} \qquad \frac{\phi;\Phi \vdash \sigma \ valid \quad \phi;\Phi \vdash \tau \ valid}{\phi;\Phi \vdash \sigma \multimap \tau \ valid}$$

$$\frac{a \notin \phi \quad (\phi, a);\Phi \vdash \Phi_a \ valid \quad (\phi, a);(\Phi, \Phi_a) \vdash \sigma_{\multimap} \ valid \quad (\phi, a);(\Phi, \Phi_a) \vDash a \leq I \text{ for some index } I}{\phi;\Phi \vdash \forall a : \Phi_a.\sigma_{\multimap} \ valid}$$

$$\frac{\phi;\Phi \vdash I \ valid \quad (\phi, a);(\Phi, a \leq I) \vdash P \ valid \quad (\phi, a);(\Phi, a \leq I) \vdash \sigma \ valid \quad \phi;\Phi \vDash \sum_{a \leq I} P = 1}{\phi;\Phi \vdash \{P : \sigma \mid a \leq I\} \ valid}$$

Fig. 3. Validity of indices, constraints and types

and constraints, and on the left subtyping correspond to finding an instantiation for the quantified variable. Finally, we add the conversion rule. Informally, this rule describe that a function of type $\mathsf{Nat}(a \mid \Phi_a) \multimap \mu$ could be given equivalently the type $\forall a : \Phi_a.Nat(a) \multimap \mu$.

Example 2.4 (Subtyping for Arrow Types). We give a common example of subtyping for arrow types: $\vdash (\forall a : a \leq I.\mathsf{Nat}(a) \multimap \mathsf{Nat}(a)) \sqsubseteq \mathsf{Nat}(b \mid b \leq I) \multimap \mathsf{Nat}(c \mid c \leq I)$.

$$\dfrac{\dfrac{\mathsf{Nat}(b \mid b \leq I) \triangleright b : (b \leq I).\mathsf{Nat}(b)} \qquad \dfrac{\dfrac{b;(b \leq I) \vDash b \leq I \quad \overline{b;(b \leq I) \vdash \mathsf{Nat}(b) \multimap \mathsf{Nat}(b) \sqsubseteq \mathsf{Nat}(b) \multimap \mathsf{Nat}(c \mid c \leq I)}}{b;(b \leq I) \vdash (\forall a : a \leq I.\mathsf{Nat}(a) \multimap \mathsf{Nat}(a)) \sqsubseteq \mathsf{Nat}(b) \multimap \mathsf{Nat}(c \mid c \leq I)}}{\vdash (\forall a : a \leq I.\mathsf{Nat}(a) \multimap \mathsf{Nat}(a)) \sqsubseteq \forall b : (b \leq I).\mathsf{Nat}(b) \multimap \mathsf{Nat}(c \mid c \leq I)}}{\vdash (\forall a : a \leq I.\mathsf{Nat}(a) \multimap \mathsf{Nat}(a)) \sqsubseteq \mathsf{Nat}(b \mid b \leq I) \multimap \mathsf{Nat}(c \mid c \leq I)}$$

And then we can conclude easily. The important point is the use of conversion rule. This conversion allows us to "extract" the elements of $\mathsf{Nat}(b \mid b \leq I)$ in order to use them in the instantiation for the $\forall - L$ rule. Notice that without conversion, there is no instantiation J of a such that $\vdash \mathsf{Nat}(b \mid b \leq I) \sqsubseteq \mathsf{Nat}(J)$

This definition of subtyping still verifies the conditions to be a preorder, as explained by the following lemma.

Lemma 2.5 (Subtyping and Preorder). *Let ϕ be an set of index variables and Φ be a set of constraints. Let ζ, ζ', ζ'' be valid types under $\phi;\Phi$. Then $\phi;\Phi \vdash \zeta \sqsubseteq \zeta$ and if $\phi;\Phi \vdash \zeta \sqsubseteq \zeta'$ and $\phi;\Phi \vdash \zeta' \sqsubseteq \zeta''$ then $\phi;\Phi \vdash \zeta \sqsubseteq \zeta''$.*

In order to work more conveniently with subtyping in the technical parts, we combine all conversion and rule for arrow types into one. This is described by the rules in Figure 5. We prove that this subtyping system and the previous one are equivalent. The only interesting cases are arrow types. First, we show how to simulate this alternative rule in the original type system. Notice that `elements` can be simulated by a chain of conversion rules and $\forall - R$ rules, as the only difference between conversion and `elements` is the fact that conversion deals with a tensor type one by one whereas `elements` does everything simultaneously.

Conversion:

$$\frac{}{\mathsf{Nat}(a \mid \Phi) \triangleright a : \Phi.\mathsf{Nat}(a)} \qquad \frac{\sigma \triangleright a : \Phi.\sigma'}{\sigma \otimes \tau \triangleright a : \Phi.(\sigma' \otimes \tau)} \qquad \frac{\tau \triangleright a : \Phi.\tau'}{\sigma \otimes \tau \triangleright a : \Phi.(\sigma \otimes \tau')}$$

Conversion Rule:

$$\frac{\sigma \triangleright a : \Phi_a.\sigma' \quad a \notin \phi \quad \phi; \Phi \vdash \tau \sqsubseteq \forall a : \Phi_a.\sigma' \multimap \mu}{\phi; \Phi \vdash \tau \sqsubseteq \sigma \multimap \mu} \text{(Conv)}$$

Coupling

$$\frac{\begin{array}{c}(\phi, a, b); (\Phi, a \le I, b \le J) \vdash S \text{ valid} \qquad (\phi, b); (\Phi, b \le J) \models \sum_{a \le I} S = Q \\ (\phi, b); (\Phi, a \le I) \models \sum_{b \le J} S = P \qquad (\phi, a, b); (\Phi, a \le I, b \le J, S \neq 0) \vdash \sigma \sqsubseteq \tau\end{array}}{\phi; \Phi \vdash S \lhd_{\sqsubseteq} \langle \{P : \sigma \mid a \le I\} \& \{Q : \tau \mid b \le J\} \rangle} \text{(Coupling)}$$

Structural Rules:

$$\frac{(\phi, a); (\Phi, \Phi_1) \models \Phi_2}{\phi; \Phi \vdash \mathsf{Nat}(a \mid \Phi_1) \sqsubseteq \mathsf{Nat}(a \mid \Phi_2)} \text{(Nat)}$$

$$\frac{\phi; \Phi \vdash \tau \sqsubseteq \sigma \quad \phi; \Phi \vdash \mu \sqsubseteq \nu}{\phi; \Phi \vdash \sigma \multimap \mu \sqsubseteq \tau \multimap \nu} (\multimap) \qquad \frac{\phi; \Phi \vdash \sigma_1 \sqsubseteq \tau_1 \quad \phi; \Phi \vdash \sigma_2 \sqsubseteq \tau_2}{\phi; \Phi \vdash \sigma_1 \otimes \sigma_2 \sqsubseteq \tau_1 \otimes \tau_2} (\otimes)$$

$$\frac{\phi; \Phi \models \Phi_a\{I/a\} \quad \phi; \Phi \vdash \sigma\{I/a\} \sqsubseteq \tau}{\phi; \Phi \vdash \forall a : \Phi_a.\sigma \sqsubseteq \tau} \text{($\forall$-L)} \qquad \frac{(\phi, a); (\Phi, \Phi_a) \vdash \sigma \sqsubseteq \tau}{\phi; \Phi \vdash \sigma \sqsubseteq \forall a : \phi_a.\tau} \text{($\forall$-R)}$$

$$\frac{\exists S, \phi; \Phi \vdash S \lhd_{\sqsubseteq} \langle \mu \& \nu \rangle}{\phi; \Phi \vdash \mu \sqsubseteq \nu} \text{(DDT)}$$

Fig. 4. Subtyping Rules

$$\frac{}{\texttt{elements}(\mathsf{Nat}(a \mid \Phi_a)) = a; \Phi_a; \mathsf{Nat}(a)} \qquad \frac{}{\texttt{elements}(\sigma_{-\circ}) = \cdot; \cdot; \sigma_{-\circ}}$$

$$\frac{\texttt{elements}(\sigma) = \phi; \Phi; \tau \qquad \texttt{elements}(\sigma') = \phi'; \Phi'; \tau' \qquad \phi \cap \phi' = \emptyset}{\texttt{elements}(\sigma \otimes \sigma') = (\phi, \phi'); (\Phi, \Phi'); (\tau \otimes \tau')}$$

$$\frac{\begin{array}{c}\texttt{elements}(\tau) = \phi_\tau; \Phi_\tau; \tau' \qquad (\phi, \overline{b}, \phi_\tau); (\Phi, \overline{\Phi}_b, \Phi_\tau) \vdash \tau' \sqsubseteq \sigma\{\overline{I}/\overline{a}\} \\ \overline{I} \text{ such that } (\phi, \overline{b}, \phi_\tau); (\Phi, \overline{\Phi}_b, \Phi_\tau) \models \overline{\Phi}_a\{\overline{I}/\overline{a}\} \qquad (\phi, \overline{b}, \phi_\tau); (\Phi, \overline{\Phi}_b, \Phi_\tau) \vdash \mu\{\overline{I}/\overline{a}\} \sqsubseteq \nu\end{array}}{\phi; \Phi \vdash \forall \overline{a} : \overline{\Phi}_a.\sigma \multimap \mu \sqsubseteq \forall \overline{b} : \overline{\Phi}_b.\tau \multimap \nu}$$

Fig. 5. Alternative Subtyping Rules

$$\frac{\texttt{elements}(\tau) = \phi_\tau; \Phi_\tau; \tau' \qquad \dfrac{(\phi, \overline{b}, \phi_\tau); (\Phi, \overline{\Phi}_b, \Phi_\tau) \vdash \sigma\{\overline{I}/\overline{a}\} \multimap \mu\{\overline{I}/\overline{a}\} \sqsubseteq \tau' \multimap \nu \qquad (\phi, \overline{b}, \phi_\tau); (\Phi, \overline{\Phi}_b, \Phi_\tau) \models \overline{\Phi}_a\{\overline{I}/\overline{a}\}}{(\phi, \overline{b}, \phi_\tau); (\Phi, \overline{\Phi}_b, \Phi_\tau) \vdash \forall \overline{a} : \overline{\Phi}_a.\sigma \multimap \mu \sqsubseteq \tau' \multimap \nu}}{\dfrac{(\phi, \overline{b}); (\Phi, \overline{\Phi}_b) \vdash \forall \overline{a} : \overline{\Phi}_a.\sigma \multimap \mu \sqsubseteq \tau \multimap \nu}{\phi; \Phi \vdash \forall \overline{a} : \overline{\Phi}_a.\sigma \multimap \mu \sqsubseteq \forall \overline{b} : \overline{\Phi}_b.\tau \multimap \nu}}$$

Now we need to prove the converse, if $\phi; \Phi \vdash \sigma_{-\circ} \sqsubseteq \tau_{-\circ}$ can be derived in the original subtype system, then it can be derived in the alternative one. We prove this by induction on the original subtype system.

- $$\frac{\tau \triangleright b : \Phi_b.\tau' \quad b \notin \phi \quad \phi;\Phi \vdash \forall \overline{a} : \overline{\Phi}_a.\sigma \multimap \mu \sqsubseteq \forall b : \Phi_b.\tau' \multimap \nu}{\phi;\Phi \vdash \forall \overline{a} : \overline{\Phi}_a.\sigma \multimap \mu \sqsubseteq \tau \multimap \nu}\ (\textsc{Conv})$$

  By induction hypothesis, we have :

  $$\frac{\begin{array}{cc}\texttt{elements}(\tau') = (b', \phi_\tau); (b' = b, \Phi_\tau); \tau'' & (\phi, b, b', \phi_\tau); (\Phi, \Phi_b, b' = b, \Phi_\tau) \vdash \tau'' \sqsubseteq \sigma\{\overline{I}/\overline{a}\} \\ \overline{I} \text{ such that } (\phi, b, b'\phi_\tau); (\Phi, b' = b, \Phi_b, \Phi_\tau) \vDash \overline{\Phi}_a\{\overline{I}/\overline{a}\} & (\phi, b, b', \phi_\tau); (\Phi, \Phi_b, b' = b, \Phi_\tau) \vdash \mu\{\overline{I}/\overline{a}\} \sqsubseteq \nu\end{array}}{\phi;\Phi \vdash \forall \overline{a} : \overline{\Phi}_a.\sigma \multimap \mu \sqsubseteq \forall b : \Phi_b.\tau' \multimap \nu}$$

  The form of $\texttt{elements}(\tau')$ can be explained by the previous conversion rule, this singleton type $\mathrm{Nat}(b)$ in τ' leads to this index variable b' and the constraint $b' = b$. Thus, in comparison, $\texttt{elements}(\tau) = (b, \phi_\tau); (\Phi_b, \Phi_\tau); \tau''\{b/b'\}$. Thus, by merging this two variables b and b', we can derive

  $$\frac{\begin{array}{cc}\texttt{elements}(\tau) = (b, \phi_\tau); (\Phi_b, \Phi_\tau); \tau'' & (\phi, b, \phi_\tau); (\Phi, \Phi_b, \Phi_\tau) \vdash \tau'' \sqsubseteq \sigma\{\overline{I}/\overline{a}\} \\ \overline{I} \text{ such that } (\phi, b, \phi_\tau); (\Phi, \Phi_b, \Phi_\tau) \vDash \overline{\Phi}_a\{\overline{I}/\overline{a}\} & (\phi, b, \phi_\tau); (\Phi, \Phi_b, \Phi_\tau) \vdash \mu\{\overline{I}/\overline{a}\} \sqsubseteq \nu\end{array}}{\phi;\Phi \vdash \forall \overline{a} : \overline{\Phi}_a.\sigma \multimap \mu \sqsubseteq \tau \multimap \nu}$$

  And this concludes this case.

- $$\frac{\phi;\Phi \vdash \tau \sqsubseteq \sigma \quad \phi;\Phi \vdash \mu \sqsubseteq \nu}{\phi;\Phi \vdash \sigma \multimap \mu \sqsubseteq \tau \multimap \nu}\ (\multimap)$$

  We can derive the proof

  $$\frac{\texttt{elements}(\tau) = \phi_\tau, \Phi_\tau; \tau' \qquad (\phi, \phi_\tau); (\Phi, \Phi_\tau) \vdash \tau' \sqsubseteq \sigma \qquad (\phi, \phi_\tau); (\Phi, \Phi_\tau) \vdash \mu \sqsubseteq \nu}{\phi;\Phi \vdash \sigma \multimap \mu \sqsubseteq \tau \multimap \nu}$$

  The weakening lemma for subtyping can be proved directly. Also, the fact that $\phi;\Phi \vdash \tau \sqsubseteq \sigma$ implies $(\phi;\phi_\tau);(\Phi,\Phi_\tau) \vdash \tau' \sqsubseteq \sigma$ can be proved directly by induction on τ and by definition of $\texttt{elements}$. This concludes this case.

- $$\frac{\phi;\Phi \vDash \Phi_a\{I/a\} \quad \phi;\Phi \vdash (\forall \overline{a} : \overline{\Phi}_a.\sigma \multimap \mu)\{I/a\} \sqsubseteq \forall \overline{b} : \overline{\Phi}_b.\tau \multimap \nu}{\phi;\Phi \vdash \forall a : \Phi_a.(\forall \overline{a} : \overline{\Phi}_a.\sigma \multimap \mu) \sqsubseteq \forall \overline{b} : \overline{\Phi}_b.\tau \multimap \nu}\ (\forall\text{-L})$$

  By induction hypothesis, we have

  $$\frac{\begin{array}{cc}\texttt{elements}(\tau) = \phi_\tau; \Phi_\tau; \tau' & (\phi, \overline{b}, \phi_\tau); (\Phi, \overline{\Phi}_b, \Phi_\tau) \vdash \tau' \sqsubseteq \sigma\{I/a\}\{\overline{I}/\overline{a}\} \\ \overline{I} \text{ such that } (\phi, \overline{b}, \phi_\tau); (\Phi, \overline{\Phi}_b, \Phi_\tau) \vDash \overline{\Phi}_a\{I/a\}\{\overline{I}/\overline{a}\} & (\phi, \overline{b}, \phi_\tau); (\Phi, \overline{\Phi}_b, \Phi_\tau) \vdash \mu\{I/a\}\{\overline{I}/\overline{a}\} \sqsubseteq \nu\end{array}}{\phi;\Phi \vdash (\forall \overline{a} : \overline{\Phi}_a.\sigma \multimap \mu)\{I/a\} \sqsubseteq \forall \overline{b} : \overline{\Phi}_b.\tau \multimap \nu}$$

  This gives us directly

  $$\frac{\begin{array}{cc}\texttt{elements}(\tau) = \phi_\tau; \Phi_\tau; \tau' & (\phi, \overline{b}, \phi_\tau); (\Phi, \overline{\Phi}_b, \Phi_\tau) \vdash \tau' \sqsubseteq \sigma\{(I, \overline{I})/(a, \overline{a})\} \\ (I, \overline{I}) \text{ such that } (\phi, \overline{b}, \phi_\tau); (\Phi, \overline{\Phi}_b, \Phi_\tau) \vDash (\Phi_a, \overline{\Phi}_a)\{(I, \overline{I})/(a, \overline{a})\} & (\phi, \overline{b}, \phi_\tau); (\Phi, \overline{\Phi}_b, \Phi_\tau) \vdash \mu\{(I, \overline{I})/(a, \overline{a})\} \sqsubseteq \nu\end{array}}{\phi;\Phi \vdash \forall (a, \overline{a}) : (\Phi_a, \overline{\Phi}_a).\sigma \multimap \mu \sqsubseteq \forall \overline{b} : \overline{\Phi}_b.\tau \multimap \nu}$$

  This concludes this case

- $$\frac{(\phi, b); (\Phi, \Phi_b) \vdash \forall \overline{a} : \overline{\Phi}_a.\sigma \multimap \mu \sqsubseteq \forall \overline{b} : \overline{\Phi}_b.\tau \multimap \nu}{\phi;\Phi \vdash \forall \overline{a} : \overline{\Phi}_a.\sigma \multimap \mu \sqsubseteq \forall b : \Phi_b.(\forall \overline{b} : \overline{\Phi}_b.\tau \multimap \nu)}\ (\forall\text{-R})$$

  By induction hypothesis we have

$$\frac{\texttt{elements}(\tau) = \phi_\tau; \Phi_\tau; \tau' \qquad (\phi, b, \overline{b}, \phi_\tau); (\Phi, \Phi_b, \overline{\Phi}_b, \Phi_\tau) \vdash \tau' \sqsubseteq \sigma\{\overline{I}/\overline{a}\} \qquad \overline{I} \text{ such that } (\phi, b, \overline{b}, \phi_\tau); (\Phi, \Phi_b, \overline{\Phi}_b, \Phi_\tau) \vDash \overline{\Phi}_a\{\overline{I}/\overline{a}\} \qquad (\phi, b, \overline{b}, \phi_\tau); (\Phi, \Phi_b, \overline{\Phi}_b, \Phi_\tau) \vdash \mu\{\overline{I}/\overline{a}\} \sqsubseteq \nu}{(\phi, b); (\Phi, \Phi_b) \vdash \forall \overline{a} : \overline{\Phi}_a.\sigma \multimap \mu \sqsubseteq \forall \overline{b} : \overline{\Phi}_b.\tau \multimap \nu}$$

And we obtain directly the derivation

$$\frac{\texttt{elements}(\tau) = \phi_\tau; \Phi_\tau; \tau' \qquad (\phi, b, \overline{b}, \phi_\tau); (\Phi, \Phi_b, \overline{\Phi}_b, \Phi_\tau) \vdash \tau' \sqsubseteq \sigma\{\overline{I}/\overline{a}\} \qquad \overline{I} \text{ such that } (\phi, b, \overline{b}, \phi_\tau); (\Phi, \Phi_b, \overline{\Phi}_b, \Phi_\tau) \vDash \overline{\Phi}_a\{\overline{I}/\overline{a}\} \qquad (\phi, b, \overline{b}, \phi_\tau); (\Phi, \Phi_b, \overline{\Phi}_b, \Phi_\tau) \vdash \mu\{\overline{I}/\overline{a}\} \sqsubseteq \nu}{\phi; \Phi \vdash \forall \overline{a} : \overline{\Phi}_a.\sigma \multimap \mu \sqsubseteq \forall (b, \overline{b}) : (\Phi_b, \overline{\Phi}_b).\tau \multimap \nu}$$

So the alternative subtyping rule is equivalent to the original one.

2.3 Typing Rules

In order to present typing rules in a clearer way, let us first give some notations on indexes. We need a way to define the convolution, that is to say to express a distribution over DDTs as a DDT. To give more intuition about that, we describe an example. Suppose given a term t of type $\{\frac{1}{i+1} : \mathrm{Nat}(a) \mid a \leq i\}$ for some integer i, and suppose that, for all $a \leq i$ and $x : \mathrm{Nat}(a)$, we can give u a type $\{\frac{1}{a+1} : \sigma \mid b \leq a\}$. Then, we would like to give $\texttt{let } x = t \texttt{ in } u$ the type $\{\frac{1}{i+1} : \{\frac{1}{a+1} : \sigma \mid b \leq a\} \mid a \leq i\}$, that is to say $\{\frac{1}{(i+1)(a+1)} : \sigma \mid a \leq i, b \leq a\}$. However, this is not in a valid form for a DDT: we need to express this using only one variable. In order to do that, we use a bijection between the sets $\{(m, m') \mid m \leq i, m' \leq m\}$ and $\{n \mid n \leq \sum_{m \leq i}(m+1)\}$, by describing the elements of the pair in the lexicographic order $(0,0); (1,0); (1;1); (2,0); \ldots$

We will now formalize this in the general case to express the convolution $\{P : \mu \mid a \leq I\}$, with $\mu = \{Q : \sigma \mid b \leq J\}$. For this, we use the following notations:

Lemma 2.6. *Let ϕ be a set of index variables and $\rho : \phi \to \mathbb{N}$ be a valuation. Let I be an index with free variables in ϕ and J an index with free variables in (ϕ, a), with $a \notin \phi$. For K with free variables in ϕ, we define $\pi_1^a(I, J, K)$ and $\pi_2^a(I, J, K)$ such that for all $\rho : \phi \to \mathbb{N}$,*
$[\![\pi_1^a(I, J, K)]\!]_\rho = sup\{m \in \mathbb{N} \mid \sum_{l<m} [\![J+1]\!]_{\rho[a \mapsto l]} \leq [\![K]\!]_\rho\}$ and $\pi_2^a(I, J, K) = K - \sum_{a < \pi_1^a(I,J,K)}(J+1)$.
We also define the operation $\star^a_{I,J}$ such that for K, K' indexes with free variables in ϕ, we have $K \star^a_{I,J} K' = \sum_{a<K}(J+1) + K'$. For a valuation $\rho : \phi \to \mathbb{N}$, those functions are bijections between the sets $\{n \mid n \leq \sum_{m \leq [\![I]\!]_\rho} [\![J+1]\!]_{\rho[a \mapsto m]}\}$ and $\{(m, m') \mid m \leq [\![I]\!]_\rho, m' \leq [\![J]\!]_{\rho[a \mapsto m]}\}$

With those notation, we can then define the convolution $\{P : \mu \mid a \leq I\}$.

Definition 2.7 (Convolution). Let $\phi; \Phi$ be a set of index variables and constraints on those variables. Let $a \notin \phi$ be an index variable, I a valid index under $\phi; \Phi$ and P a valid rational index under $(\phi, a); (\Phi, a \leq I)$ such that $\phi; \Phi \vDash \sum_{a \leq I} P = 1$. Let $\mu = \{Q : \sigma \mid b \leq J\}$ be a valid DDT under $(\phi, a); (\Phi, a \leq I)$. We define the *convolution* $\{P : \mu \mid a \leq I\}$ as the DDT
$\nu = \{(P \cdot Q)\{\pi_1^a(I, J, c)/a\}\{\pi_2^a(I, J, c)/b\} : \sigma\{\pi_1^a(I, J, c)/a\}\{\pi_2^a(I, J, c)/b\} \mid c \leq \sum_{a \leq I}(J+1)\}$.

We can now describe the type system for our calculus. For this we introduce two kind of variables contexts : linear and valuation contexts.

Definition 2.8 (Linear Contexts). A *linear context* Γ is a non-ordered sequence $\Gamma = x_1 : \sigma_1, \ldots x_n : \sigma_n$. We usually write a context $\ell\Gamma$ when all those types are Nat types. For two type contexts Γ and Δ, we denote the concatenation of those contexts by Γ, Δ. This concatenation is defined if and only if for each variable x with $(x : \sigma) \in \Gamma$ and $(x : \tau) \in \Delta$, then $\sigma = \tau = \mathrm{Nat}(a \mid \Phi)$ for some Φ.

Definition 2.9 (Valuation Contexts). A *valuation context* Θ is either empty or a context of the form $y : \{P : \sigma \mid a \leq I\}$. We keep the syntactic definition of DDT but the only difference is that for validity we do not enforce any condition on the sum of P: it can be 0 or more than 1 in particular,

thus this represents a valuation and not a distribution The intuition about $y : \{P : \sigma \mid a \leq I\}$ is that P is the expected number of calls to y with type σ. We can then adapt definitions for DDTs such as couplings, subtyping and convolution to a valuation context. The concatenation $\Theta + \Psi$ of two valuation contexts Θ and Ψ is defined only if $\Theta = y : \{P : \sigma \mid a \leq I\}$, $\Psi = y : \{Q : \sigma \mid a \leq I\}$, and then $\Theta + \Psi = y : \{P + Q : \sigma \mid a \leq I\}$.

Typing judgments have the form $\phi; \Phi; \Gamma \mid \Theta \vdash_R t : \mu$, with all types present in Γ, Θ and μ valid under $\phi; \Phi$. Active terms are typed with a distribution type μ and values are typed with a linear dependent type σ. The rational index R is called the *weight*. It is an indication of the expected runtime of a term and will be used for type soundness.

The type system is given in Figure 6. The axiom rule for recursion context follows the intuition given previously: if the expected number of call to y with type $\sigma\{J/a\}$ is greater than 1, then y can be given the type $\sigma\{J/a\}$. In the rule for the introduction of $\forall$, the maximum is well-defined and finite since all universal quantification are bounded, as expressed by the definition of validity. Another interesting rule is the one for pattern matching. The typing of the integer and the first case is quite intuitive, but for the second one, in the set of constraints Φ', a is not free in Φ' (Φ' must be valid under $(\phi; \Phi)$). So, $(\phi, a); (\Phi, \Phi_a\{a + 1/a\}) \vDash \Phi'$ express that Φ' is a set of constraint deduced form the fact that at least one integer different from 0 satisfies Φ_a. A detailed example will be given later. Finally, in the rule for the fixpoint, the different recursive calls to y are expressed by the valuation type $\{P : \sigma_{\multimap}\{M/b\} \mid a \leq I\}$, M being a natural index. And with this description of recursive calls, we ask for an index Q that satisfies the recurrence relation derived from those recursive calls. This will also be explained in an example.

2.4 Example

We conclude this section with a simple example demonstrating the use of our type system. Let us consider again the term for biased random walk:

$$\mathtt{rwalk} \triangleq \mathtt{fix}\ rw.\lambda n.\mathtt{match}\ n\ \mathtt{with}\ \{\ \mathtt{z} \mapsto\ \mathtt{z} \mid \mathtt{s} \mapsto\ \lambda n'.rw\ (\mathtt{s}(\mathtt{s}(n'))) \oplus_2 rw\ n'\ \}\ .$$

We show that we can give this term $\mathtt{rwalk}$ the following type : $\vdash \mathtt{rwalk} : \mathrm{Nat}(J) \multimap \mathrm{Nat}(0)$. For the sake of simplicity, we do not detail the weight of this proof, but we will talk briefly about it later. We define the function *Bool* such that $Bool(0) = 0$ and $Bool(n + 1) = 1$. This function is very useful to present index in a simpler form. The first important rule is the one for fixpoint.

$$\frac{b; \top; \cdot \mid rw : \{\frac{Bool(b)\cdot(a+1)}{3} : \mathrm{Nat}(b + 1 - 2a) \multimap \mathrm{Nat}(0) \mid a \leq 1\} \vdash \lambda n \ldots : \mathrm{Nat}(b) \multimap \mathrm{Nat}(0)}{\vdash \mathtt{fix}\ rw.\lambda n \ldots : (\mathrm{Nat}(b) \multimap \mathrm{Nat}(0))\{J/b\}}$$

Let us describe this valuation context. When $b = 0$, that is to say when we are in the case $\mathtt{rwalk}\ \underline{0}$, this valuation context expresses that we never call rw. So it means that we stop the recursion. We can see here a reason why we consider valuation and not distribution. Otherwise, when $b \neq 0$, (that is to say in the case $\mathtt{rwalk}\ \underline{n}$ with $n > 0$) this valuation context represents the distribution $rw : \{\frac{1}{3} : \mathrm{Nat}(b + 1) \multimap \mathrm{Nat}(0); \frac{2}{3} : \mathrm{Nat}(b - 1) \multimap \mathrm{Nat}(0)\}$. So it expresses that with probability $\frac{1}{3}$, $\mathtt{rwalk}$ will be called with the input $\underline{n + 1}$ and with probability $\frac{2}{3}$ it will be called with the input $\underline{n - 1}$. One can remark that in this case, the index M used in the fixpoint describes the reachable states from b in the Markov chain describing this random walk. From now on, let us call Θ the valuation context $rw : \{\frac{Bool(b)\cdot(a+1)}{3} : \mathrm{Nat}(b + 1 - 2a) \multimap \mathrm{Nat}(0) \mid a \leq 1\}$. We now give the typing for the $\mathtt{match}$ rule:

$$\frac{\begin{array}{cc} b; (0 = b); \cdot \mid \Theta \vdash \mathtt{z} : \mathrm{Nat}(0) & (b, a); (a + 1 = b) \vDash (b \geq 1) \\ b; \top; n : \mathrm{Nat}(b) \mid \cdot \vdash n : \mathrm{Nat}(a \mid a = b) & b; (b \geq 1); \cdot \mid \Theta \vdash \lambda n' \cdots : \mathrm{Nat}(a \mid a + 1 = b) \multimap \mathrm{Nat}(0) \end{array}}{b; \top; n : \mathrm{Nat}(b) \mid \Theta \vdash \mathtt{match}\ n\ \mathtt{with}\ \{\ \mathtt{z} \mapsto\ \mathtt{z} \mid \mathtt{s} \mapsto\ \lambda n' \ldots\ \} : \mathrm{Nat}(0)}$$

$$\frac{}{\phi;\Phi;\Gamma,x:\sigma \mid \Theta \vdash_0 x:\sigma} \qquad \frac{\phi;\Phi \vDash P\{J/a\} \geq 1 \qquad \phi;\Phi \vDash J \leq I}{\phi;\Phi;\Gamma \mid y:\{P:\sigma \mid a \leq I\} \vdash_0 y:\sigma\{J/a\}}$$

$$\frac{\phi;\Phi \vDash \Phi_a\{0/a\}}{\phi;\Phi;\Gamma \mid \Theta \vdash_0 \mathsf{z}:\mathsf{Nat}(a \mid \Phi_a)} \qquad \frac{\phi;\Phi;\Gamma \mid \Theta \vdash_R v:\mathsf{Nat}(a \mid \Phi_a)}{\phi;\Phi;\Gamma \mid \Theta \vdash_R \mathsf{s}(v):\mathsf{Nat}(a \mid \Phi_a\{a-1/a\}, a \neq 0)}$$

$$\frac{}{\phi;\Phi;\Gamma \mid \Theta \vdash_1 \mathtt{Unif}:\mathsf{Nat}(I) \multimap \{\tfrac{1}{I+1}:\mathsf{Nat}(a) \mid a \leq I\}} \qquad \frac{\phi;\Phi;\Gamma,x:\sigma \mid \Theta \vdash_R t:\mu}{\phi;\Phi;\Gamma \mid \Theta \vdash_{R+1} \lambda x.t:\sigma \multimap \mu}$$

$$\frac{\phi;\Phi;\Gamma \mid \Theta \vdash_R v:\sigma \multimap \mu \qquad \phi;\Phi;\Delta \mid \Psi \vdash_{R'} w:\sigma}{\phi;\Phi;\Gamma,\Delta \mid \Theta+\Psi \vdash_{R+R'} v\ w:\mu} \qquad \frac{\phi;\Phi;\Gamma \mid \Theta \vdash_R v:\sigma \qquad \phi;\Phi;\Delta \mid \Psi \vdash_{R'} w:\tau}{\phi;\Phi;\Gamma,\Delta \mid \Theta+\Psi \vdash_{R+R'} \langle v,w\rangle:\sigma \otimes \tau}$$

$$\frac{(\phi,a);(\Phi,\Phi_a);\Gamma \mid \Theta \vdash_R v:\sigma_{\multimap}}{\phi;\Phi;\Gamma \mid \Theta \vdash_{max_{a:\Phi_a}(R)} v:\forall a:\Phi_a.\sigma_{\multimap}} \qquad \frac{\phi;\Phi;\Gamma \mid \Theta \vdash_R v:\forall a:\Phi_a.\sigma_{\multimap} \qquad \phi;\Phi \vDash \Phi_a\{I/a\}}{\phi;\Phi;\Gamma \mid \Theta \vdash_R v:\sigma_{\multimap}\{I/a\}}$$

$$\frac{\phi;\Phi;\Gamma \mid \Theta \vdash_R v:\sigma \otimes \tau \qquad \phi;\Phi;\Delta,x:\sigma,y:\tau \mid \Psi \vdash_{R'} t:\mu}{\phi;\Phi;\Gamma,\Delta \mid \Theta+\Psi \vdash_{1+R+R'} \mathtt{let}\ \langle x,y\rangle = v\ \mathtt{in}\ t:\mu}$$

$$\frac{\phi;\Phi;\Gamma \mid \Theta \vdash_R t:\nu \qquad \phi;\Phi \vdash (\Delta \mid \Psi) \sqsubseteq (\Gamma \mid \Theta) \qquad \phi;\Phi \vdash \nu \sqsubseteq \mu \qquad \phi;\Phi \vDash R \leq R'}{\phi;\Phi;\Delta \mid \Psi \vdash_{R'} t:\mu}$$

$$\frac{\begin{array}{ll}\phi;(\Phi,\Phi_a\{0/a\});\Delta \mid \Psi \vdash_R t:\mu & (\phi,a);(\Phi,\Phi_a\{a+1/a\}) \vDash \Phi' \\ \phi;\Phi;\Gamma \mid \Theta \vdash_{R'} v:\mathsf{Nat}(a \mid \Phi_a) & \phi;(\Phi,\Phi');\Delta \mid \Psi \vdash_R w:\mathsf{Nat}(a \mid \Phi_a\{a+1/a\}) \multimap \mu\end{array}}{\phi;\Phi;\Gamma,\Delta \mid \Theta+\Psi \vdash_{1+R+R'} \mathtt{match}\ v\ \mathtt{with}\ \{\ \mathsf{z} \mapsto\ t \mid \mathsf{s} \mapsto\ w\ \}:\mu}$$

$$\frac{\phi;\Phi;\Gamma \mid \Theta \vdash_{R'} t:\{P:\sigma \mid a \leq I\} \qquad (\phi,a);(\Phi,a \leq I,P \neq 0);\Delta,x:\sigma \mid y:\nu \vdash_R u:\mu}{\phi;\Phi;\Gamma,\Delta \mid \Theta+(y:\{P:\nu \mid a \leq I\}) \vdash_{1+R'+\sum_{a\leq I,P\neq 0} P\cdot R} \mathtt{let}\ x=t\ \mathtt{in}\ u:\{P:\mu \mid a \leq I\}}$$

$$\frac{\begin{array}{c}(\phi,b);\Phi \vDash Q \geq 1+R+\sum_{a\leq I,P\neq 0} P\cdot Q\{M/b\} \\ (\phi,b);\Phi;\ell\Gamma \mid y:\{P:\sigma_{\multimap}\{M/b\} \mid a \leq I\} \vdash_R v:\sigma_{\multimap}\end{array}}{\phi;\Phi;\Gamma,\ell\Gamma \mid \Theta \vdash_{Q\{J/b\}} \mathtt{fix}\ y.v:\sigma_{\multimap}\{J/b\}}$$

Fig. 6. Type System for dℓRPCF

The variable n has type $\mathsf{Nat}(b)$. Thus, when we know that n is a successor, in the second branch of the match, we can prove that $b \geq 1$. This is expressed by $(b,a);(a+1=b) \vDash (b \geq 1)$. So we can use this hypothesis $(b \geq 1)$ in the second branch of the `match`. Remark also that when $b \geq 1$, the type $\mathsf{Nat}(a \mid a+1=b)$ is equivalent to $\mathsf{Nat}(b-1)$.

For the sake of simplicity, we now show informally how to use the probabilistic choice $\oplus_2$. This operator (or more precisely the `let` operator) allow us to "split" the valuation context according to the probabilities of each branch of this probabilistic choice. Thus, the valuation context Θ can be split into two valuation context Ψ_1 and Ψ_2 if Θ correspond to the convolution $\{\frac{1}{3}:\Psi_1;\frac{2}{3}:\Psi_2\}$. As Θ represents $rw:\{\frac{1}{3}:\mathsf{Nat}(b+1) \multimap \mathsf{Nat}(0);\frac{2}{3}:\mathsf{Nat}(b-1) \multimap \mathsf{Nat}(0)\}$ we can take
$\Psi_1 = rw:\{1:\mathsf{Nat}(b+1) \multimap \mathsf{Nat}(0)\}$ and $\Psi_2 = rw:\{1:\mathsf{Nat}(b-1) \multimap \mathsf{Nat}(0)\}$ used to type $rw\ (\mathsf{s}(\mathsf{s}(n')))$ and $rw\ n'$ respectively. Formally, the intuition given above correspond to a subtyping rule for valuation context.

Finally, for the weight of this proof, the subterm

$$\lambda n.\mathtt{match}\ n\ \mathtt{with}\ \{\ \mathsf{z} \mapsto\ \mathsf{z} \mid \mathsf{s} \mapsto\ \lambda n'.rw\ (\mathsf{s}(\mathsf{s}(n'))) \oplus_2 rw\ n'\ \}$$

can be typed with a constant weight equals to 7. Thus, the fixpoint rule gives us the inequation $b; \top \vDash Q \geq 8 + \sum_{a \leq 1} \frac{Bool(b) \cdot (a+1)}{3} \cdot Q\{b + 1 - 2a/b\}$. That is to say,
$\vDash Q\{0/b\} \geq 8$ and $b; (b \geq 1) \vDash Q \geq 8 + \frac{1}{3} \cdot Q\{b + 1/b\} + \frac{2}{3} \cdot Q\{b - 1/b\}$. This relation is thus very close to the expression of the expected time of a random walk. A solution of this inequation is $Q = 8(3b + 1)$.

3 TYPE SOUNDNESS

We now show the type soundness of our calculus. For this, we show that the weight described in the type system in Figure 6 is a bound on the expected runtime of a term. As we proved that they are equivalent, we work with the alternative subtyping rule.

3.1 Weakening, Contraction and Index Substitution

We first present general proprieties on indexes, types and typing derivation. Especially, we show that the use of a set of index variable ϕ and a set of constraint Φ corresponds intuitively to a universal quantification over all variable ϕ that satisfies Φ.

Lemma 3.1 (Weakening). *Let $\phi; \Phi$ be a set of index variables and constraints. Let ϕ' be a set of index variables disjoint from ϕ. Let Φ' be a set of constraint valid under $(\phi, \phi'); \Phi$.*

1. *If $\phi; \Phi \vDash A \mathcal{R} B$ then $(\phi, \phi'); (\Phi, \Phi') \vDash A \mathcal{R} B$.*
2. *If $\phi; \Phi \vdash \zeta$ valid then $(\phi, \phi'); \Phi, \Phi' \vdash \zeta$ valid.*
3. *If $\phi; \Phi \vdash \zeta \sqsubseteq \xi$ then $(\phi, \phi'); (\Phi, \Phi') \vdash \zeta \sqsubseteq \xi$.*
4. *If $\phi; \Phi; \Gamma \mid \Theta \vdash_R t : \mu$ then for all ϕ', Φ', Γ' such that the concatenation Γ, Γ' are defined, we have $(\phi, \phi'); (\Phi, \Phi'); \Gamma, \Gamma' \mid \Theta \vdash_R t : \mu$. Moreover, if $\Theta = \emptyset$ then for any Θ' we have $(\phi, \phi'); (\Phi, \Phi'); \Gamma, \Gamma' \mid \Theta' \vdash_R t : \mu$*

The first point comes from the definition of $\vDash$. Then, points 2,3 and 4 are proved by induction on the definition of validity, definition of subtyping and typing derivation. All cases are direct.

Lemma 3.2 (Constraint Contraction). *Let $(\phi; \Phi)$ be a set of index variables and constraints. Let C be a constraint on ϕ such that $\phi; \Phi \vDash C$.*

1. *If $\phi; (\Phi, C) \vDash A \mathcal{R} B$ then $\phi; \Phi \vDash A \mathcal{R} B$.*
2. *If $\phi; (\Phi, C) \vdash \zeta$ valid then $\phi; \Phi \vdash \zeta$ valid.*
3. *If $\phi; (\Phi, C) \vdash \zeta \sqsubseteq \xi$ then $\phi; \Phi \vdash \zeta \sqsubseteq \xi$.*
4. *If $\phi; (\Phi, C); \Gamma \mid \Theta \vdash_R t : \mu$ then $\phi; \Phi; \Gamma \mid \Theta \vdash_R t : \mu$.*

Again, the first point comes from the definition of $\vDash$ and the other ones are proved by induction. All cases are rather direct using the weakening lemma. This shows that set of constraints work as expected: if a constraint does not add any information, then it can be removed. Finally, we show that we can instantiate index variables in ϕ.

Lemma 3.3 (Index Substitution). *Let $(\phi, a); \Phi$ be a set of index variables and constraints. Let K be a natural index with free variables in ϕ.*

1. *For a valuation $\rho : \phi \to \mathbb{N}$, we have for all index A with free variables in (ϕ, a), $[\![A\{K/a\}]\!]_\rho = [\![A]\!]_{\rho[a \mapsto [\![K]\!]_\rho]}$. Moreover, $\rho \vDash \Phi\{K/a\}$ iff $\rho[a \mapsto [\![K]\!]_\rho] \vDash \Phi$.*
2. *If $(\phi, a); \Phi \vDash A \mathcal{R} B$ then $\phi; \Phi\{K/a\} \vDash (A\{K/a\}) \mathcal{R} (B\{K/a\})$.*
3. *If $(\phi, a); \Phi \vdash \zeta$ valid then $\phi; \Phi\{K/a\} \vdash \zeta\{K/a\}$ valid.*
4. *If $(\phi, a); \Phi \vdash \zeta \sqsubseteq \xi$ then $\phi; \Phi\{K/a\} \vdash \zeta\{K/a\} \sqsubseteq \xi\{K/a\}$.*
5. *If $(\phi, a); \Phi; \Gamma \mid \Theta \vdash_R t : \mu$ then $\phi; \Phi\{K/a\}; \Gamma\{K/a\} \mid \Theta\{K/a\} \vdash_{R\{K/a\}} t : \mu\{K/a\}$.*

The first point comes from the definition of $[\![\cdot]\!]_\rho$. Then, the other points are proved by induction. This is rather direct using the fact that, for all index I with free variables in (ϕ, a, b), J with free variable in (ϕ, a) and K with free variable in ϕ, we have $I\{J/b\}\{K/a\} = I\{K/a\}\{J\{K/a\}/b\}$.

3.2 Proprieties of Values and Substitution Lemmas

We now present the usual substitution lemmas in the case of our calculus. In order to do that, we show the rule of values in our calculus.

Lemma 3.4 (Progress). *Let $\phi; \Phi; \cdot \mid \cdot \vdash_R t : \mu$. Then t is normal for $\rightarrow$ if and only if t is a value v.*

This is not surprising as it correspond exactly to the Lemma 1.1 in the simple type system. A particular kind of values that interest us are integers values and their type. We can show that the type $\mathsf{Nat}(a \mid \Phi_a)$ is indeed the set of integers n such that $\Phi_a\{n/a\}$.

Lemma 3.5 (Integers Values). *Let v be a value such that $\phi; \Phi; \cdot \mid \cdot \vdash_R v : \mathsf{Nat}(a \mid \Phi_a)$, then $v = \underline{n}$ for some integer n and $\phi; \Phi \vDash \Phi_a\{n/a\}$. Reciprocaly, for any n such that $\phi; \Phi \vDash \Phi_a\{n/a\}$, we can give the typing $\phi; \Phi; \cdot \mid \cdot \vdash_0 \underline{n} : \mathsf{Nat}(a \mid \Phi_a)$.*

This is proved by induction on the typing rule for the first proposition. The second proposition is rather direct. Then, we work on the definition of `elements`. Intuitively, `elements` describe the elements of a type, and subtyping correspond to inclusion between types. Thus, we obtain the following lemma:

Lemma 3.6 (Subtyping and Elements). *Let σ, τ be valid types under $\phi; \Phi$ such that $\phi; \Phi \vdash \sigma \sqsubseteq \tau$. Then, we have $\mathtt{elements}(\sigma) = \phi'; \Phi_\sigma; \sigma'$ and $\mathtt{elements}(\tau) = \phi'; \Phi_\tau; \tau'$ such that $(\phi, \phi'); (\Phi, \Phi_\sigma) \vDash \Phi_\tau$ and $(\phi, \phi'); (\Phi, \Phi_\sigma) \vdash \sigma' \sqsubseteq \tau'$.*

The fact that we can give the same set of index variable ϕ' for the two `elements` is just a matter of renaming, as σ and τ have the same form by subtyping. We prove this by induction on σ (and τ). If σ is an arrow type, this is direct by definition of `elements`. If σ is a tensor type, this is rather direct by induction hypothesis, by definition of subtyping for tensor and by weakening. If σ is $\mathsf{Nat}(a \mid \Phi_a)$ for some Φ_a, then this is exactly the definition of subtyping of integers. We can now show the link between `elements` and typed values.

Lemma 3.7 (values and Elements). *Let v be a value such that $\phi; \Phi; \cdot \mid \cdot \vdash_R v : \sigma$. Suppose that $\mathtt{elements}(\sigma) = \overline{a}; \Phi_a; \tau$. Then there exists an instantiation $\overline{I}$ of $\overline{a}$ valid under $\phi; \Phi$ such that $\phi; \Phi \vDash \Phi_a\{\overline{I}/\overline{a}\}$ and $\phi; \Phi; \cdot \mid \cdot \vdash_R v : \tau\{\overline{I}/\overline{a}\}$.*

We prove this by induction on the type σ. If σ is an arrow type then this is direct by definition of `elements`. Let us detail the two other cases.

- Nat: If $\sigma = \mathsf{Nat}(a \mid \Phi_a)$. By definition, $\mathtt{elements}(\sigma) = a; \Phi_a; \mathsf{Nat}(a)$. By Lemma 3.5, $v = \underline{n}$ for some integer n and $\phi; \Phi \vDash \Phi_a\{n/a\}$. Thus, we can take n for instantiation of a and again by Lemma 3.5, as $\phi; \Phi \vDash (a' = n)\{n/a'\}$, we can give the typing $\phi; \Phi; \cdot \mid \cdot \vdash_0 \underline{n} : \mathsf{Nat}(a)\{n/a\}$. This concludes the case.
- Tensor: If $\sigma = \sigma_0 \otimes \sigma_1$, then the typing $\phi; \Phi; \cdot \mid \cdot \vdash_R v : \sigma_0 \otimes \sigma_1$ can be given a particular form. Indeed, by reflexivity and transitivity of subtyping, any chain of subtyping rule is equivalent to exactly one subtyping rule. As σ is not an arrow type, subtyping is the only possible non syntax-directed rule, and the only way to give a value a tensor type is if this value is a tensor of value and by using a tensor introduction rule. Thus, we have the typing

$$\dfrac{\dfrac{\phi; \Phi; \cdot \mid \cdot \vdash_{R_0} v_0 : \tau_0 \qquad \phi; \Phi; \cdot \mid \cdot \vdash_{R_1} v_1 : \tau_1}{\phi; \Phi; \cdot \mid \cdot \vdash_{R_0 + R_1} \langle v_0, v_1 \rangle : \tau_0 \otimes \tau_1} \qquad \begin{array}{c} \phi; \Phi \vDash R_0 + R_1 \leq R \\ \phi; \Phi \vdash \tau_0 \otimes \tau_1 \sqsubseteq \sigma_0 \otimes \sigma_1 \end{array}}{\phi; \Phi; \cdot \mid \cdot \vdash_R v = \langle v_0, v_1 \rangle : \sigma_0 \otimes \sigma_1}$$

Let us pose $\texttt{elements}(\tau_0) = \overline{a}_0; \Phi_0; \tau_0'$ and $\texttt{elements}(\tau_1) = \overline{a}_1; \Phi_1; \tau_1'$. By induction hypothesis, there exists instantiations $\overline{I}_0$ of $\overline{a}_0$ and $\overline{I}_1$ of $\overline{a}_1$ such that $\phi; \Phi \vDash \Phi_0\{\overline{I}_0/\overline{a}_0\}$ and $\phi; \Phi \vDash \Phi_1\{\overline{I}_1/\overline{a}_1\}$ and $\phi; \Phi; \cdot \mid \cdot \vdash_{R_0} v_0 : \tau_0'\{\overline{I}_0/\overline{a}_0\}$ and $\phi; \Phi; \cdot \mid \cdot \vdash_{R_1} v_1 : \tau_1'\{\overline{I}_1/\overline{a}_1\}$. By Lemma 3.6, we have $\texttt{elements}(\sigma_0 \otimes \sigma_1) = (\overline{a}_0, \overline{a}_1); \Phi_\sigma; \sigma'$ such that $(\phi, \overline{a}_0, \overline{a}_1); (\Phi, \Phi_0, \Phi_1) \vDash \Phi_\sigma$ and $(\phi, \overline{a}_0, \overline{a}_1); (\Phi, \Phi_0, \Phi_1) \vdash \tau_0' \otimes \tau_1' \sqsubseteq \sigma'$. Thus, by Lemma 3.3 and Lemma 3.2, we obtain $\phi; \Phi \vDash \Phi_\sigma\{\overline{I}_0, \overline{I}_1/\overline{a}_0, \overline{a}_1\}$ and $\phi; \Phi \vdash (\tau_0' \otimes \tau_1')\{\overline{I}_0, \overline{I}_1/\overline{a}_0, \overline{a}_1\} \sqsubseteq \sigma'\{\overline{I}_0, \overline{I}_1/\overline{a}_0, \overline{a}_1\}$. By subtyping, from the proof $\phi; \Phi; \cdot \mid \cdot \vdash_{R_0+R_1} \langle v_0, v_1\rangle : (\tau_0' \otimes \tau_1')\{\overline{I}_0, \overline{I}_1/\overline{a}_0, \overline{a}_1\}$ we can obtain the proof $\phi; \Phi; \cdot \mid \cdot \vdash_R \langle v_0, v_1\rangle : \sigma'\{\overline{I}_0, \overline{I}_1/\overline{a}_0, \overline{a}_1\}$, and this concludes the case.

We can now present the substitution lemmas for our calculus. In our reduction rules, the only terms that are used in a substitution are values, thus we can restrict substitution lemmas to values. There are two kind of substitution in this calculus. First, a standard substitution corresponding to non-monadic types.

Lemma 3.8 (Substitution Lemma for Linear Contexts). *Suppose that we have a proof $\phi; \Phi; \Gamma, x : \sigma \mid \Theta \vdash_R t : \mu$ and a proof $\phi; \Phi; \cdot \mid \cdot \vdash_{R'} v : \sigma$ then we can derive the proof $\phi; \Phi; \Gamma \mid \Theta \vdash_{R+R'} t[x := v] : \mu$*

The lemma is essentially a consequence of linearity of the type system. In order to prove this we distinguish two cases. If $\sigma = \mathrm{Nat}(a \mid \Phi_a)$ for some Φ_a, then x is duplicable. However, by the previous lemma, $v = \underline{n}$ for some $n \in \mathbb{N}$. So we have a proof $\phi; \Phi; \cdot \mid \cdot \vdash_0 v : \mathrm{Nat}(a \mid \Phi_a)$. This weight equals to 0 implies that we have $\phi; \Phi; \Gamma \mid \Theta \vdash_R t[x := v] : \mu$. Otherwise, $x : \sigma$ appears in only one context in a concatenation Γ, Δ and we can use the induction hypothesis on the branches where x appears.

The other substitution is the one for valuation contexts, in which variable are give a dynamic valuation type. For this let us first give a notation for indexes that we will use in the proof.

Definition 3.9 (Extract). Let ϕ be a set of index variables, $a \notin \phi$, and I an index with free variables in ϕ and J an index with free variable in (ϕ, a). We define $ext(a, I, J)$ as an index with free variables in ϕ such that $\phi; \sum_{a \le I} J \neq 0 \vDash (ext(a, I, J) \le I) \wedge (J\{ext(a, I, J)/a\} \neq 0)$ and $(\phi, a); (a \le I, J = 0) \vDash ext(a, I, J) = 0$. Intuitively, for $\rho : \phi \to \mathbb{N}$, when $[\![\sum_{a \le I} J]\!]_\rho$ is non-null, then $[\![ext(a, I, J)]\!]_\rho$ is an integer n smaller than $[\![I]\!]_\rho$ such that $[\![J]\!]_{\rho[a \mapsto n]}$ is non-null. Note that with this definition, there is not a unique choice for $[\![ext(a, I, J)]\!]_\rho$, but when we do not say otherwise, we just take the smaller integer verifying those conditions.

Lemma 3.10 (Substitution Lemma for Valuation Contexts). *Suppose that $\phi; \Phi; \Gamma \mid y : \{P : \sigma \mid a \le I\} \vdash_{R'} t : \mu$ and $(\phi, a); (\Phi, a \le I, P \neq 0); \cdot \mid \cdot \vdash_R v : \sigma$. Then $\phi; \Phi; \Gamma \mid \cdot \vdash_{R' + \sum_{a \le I, P \neq 0} P \cdot R} t[y := v] : \mu$.*

Proof. We prove this by induction on the typing of t. We detail some non-trivial cases.

- Axiom: Suppose we have the typing :
$$\frac{\phi; \Phi \vDash P\{J/a\} \ge 1 \qquad \phi; \Phi \vDash J \le I}{\phi; \Phi; \Gamma \mid y : \{P : \sigma \mid a \le I\} \vdash_0 y : \sigma\{J/a\}}$$
And suppose $(\phi, a); (\Phi, a \le I, P \neq 0); \cdot \mid \cdot \vdash_R v : \sigma$. By Lemma 3.3, we obtain $\phi; (\Phi, J \le I, P\{J/a\} \neq 0); \cdot \mid \cdot \vdash_{R\{J/a\}} v : \sigma\{J/a\}$. As $\phi; \Phi \vDash J \le I$ and $\phi; \Phi \vDash P\{J/a\} \ge 1$, by Lemma 3.2, we obtain $\phi; \Phi; \cdot \mid \cdot \vdash_{R\{J/a\}} v : \sigma\{J/a\}$. By weakening, we obtain $\phi; \Phi; \Gamma \mid \cdot \vdash_{R\{J/a\}} y[y := v] : \sigma\{J/a\}$. As $\phi; \Phi \vDash 1 \le P\{J/a\}$ and $\phi; \Phi \vDash J \le I$, we have $\phi; \Phi \vDash R\{J/a\} \le (P \cdot R)\{J/a\} \le \sum_{a \le I, P \neq 0} P \cdot R$ and we can conclude this case by subtyping.
- Application: Suppose we have the typing :

$$\frac{\phi;\Phi;\Gamma \mid y : \{P : \sigma \mid a \le I\} \vdash_{R_0} w_0 : \sigma \multimap \mu \qquad \phi;\Phi;\Delta \mid y : \{Q : \sigma \mid a \le I\} \vdash_{R_1} w_1 : \sigma}{\phi;\Phi;\Gamma,\Delta \mid y : \{(P+Q) : \sigma \mid a \le I\} \vdash_{R_0+R_1} w_0\ w_1 : \mu}$$

And suppose $(\phi, a);(\Phi, a \le I, P+Q \neq 0);\cdot \mid \cdot \vdash_R v : \sigma$. By weakening, we obtain $(\phi, a);(\Phi, a \le I, P+Q \neq 0, P \neq 0);\cdot \mid \cdot \vdash_R v : \sigma$, and then by contraction, we obtain $(\phi, a);(\Phi, a \le I, P \neq 0);\cdot \mid \cdot \vdash_R v : \sigma$. By induction hypothesis, we have $\phi;\Phi;\Gamma \mid \cdot \vdash_{R_0+\sum_{a\le I,P\neq 0} P\cdot R} w_0[y := v] : \sigma \multimap \mu$. In the same way, we obtain $\phi;\Phi;\Delta \mid \cdot \vdash_{R_1+\sum_{a\le I,Q\neq 0} Q\cdot R} w_1[y := v] : \sigma$. Thus, by using the application rule, we obtain $\phi;\Phi;\Gamma,\Delta \mid \cdot \vdash_{R_0+R_1+\sum_{a\le I,(P+Q)\neq 0}(P+Q)\cdot R} (w_0\ w_1)[y := v] : \mu$. The same method can be used for any rule that uses this concatenation $\Theta + \Psi$ such as tensor introduction, tensor elimination and pattern matching.

- Subtyping : Suppose we have the following derivation tree :

$$\frac{\dfrac{\phi;\Phi;\Delta \mid y : \{Q : \tau \mid b \le J\} \vdash_{R_0} t : \nu \qquad \phi;\Phi \vdash \{P : \sigma \mid a \le I\} \sqsubseteq \{Q : \tau \mid b \le J\}}{\phi;\Phi;\Gamma \mid y : \{P : \sigma \mid a \le I\} \vdash_{R'} t : \mu} \qquad \phi;\Phi \vdash \nu \sqsubseteq \mu, \Gamma \sqsubseteq \Delta \qquad \phi;\Phi \vDash R_0 \le R_1}$$

And we have $(\phi, a);(\Phi, a \le I, P \neq 0);\cdot \mid \cdot \vdash_R v : \sigma$. By definition of subtyping, there exists a rational index S such that $\phi;\Phi \vdash S \lhd_{\sqsubseteq} \langle\{P : \sigma \mid a \le I\}\&\{Q : \tau \mid b \le J\}\rangle$. By weakening, we obtain the proof $(\phi, a, b);(\Phi, a \le I, P \neq 0, b \le J, Q \neq 0, S \neq 0);\cdot \mid \cdot \vdash_R v : \sigma$. By subtyping, we have $(\phi, a, b);(\Phi, a \le I, P \neq 0, b \le J, Q \neq 0, S \neq 0);\cdot \mid \cdot \vdash_R v : \tau$. Let us call $H = ext(a, I, S)$, with H chosen such that $R\{H/a\}$ is minimal. By lemma 3.3, we obtain : $(\phi, b);(\Phi, H \le I, P\{H/a\} \neq 0, b \le J, Q \neq 0, S\{H/a\} \neq 0);\cdot \mid \cdot \vdash_{R\{H/a\}} v : \tau$. Moreover, by definition we have $(\phi, b);(Q \neq 0) \vDash (H \le I) \wedge (S\{H/a\} \neq 0)$.
As $(\phi, a, b);(\Phi, a \le I, b \le J, S \neq 0) \vDash P \neq 0$, we also have $(\phi, b);(\Phi, H \le I, b \le J, S\{H/a\} \neq 0) \vDash P\{H/a\} \neq 0$. Thus, by contraction, we obtain : $(\phi, b);(\Phi, b \le J, Q \neq 0);\cdot \mid \cdot \vdash_{R\{H/a\}} v : \tau$. Moreover, $(\phi, b);(\Phi, b \le J, Q \neq 0) \vDash R\{H/a\} \le \sum_{a\le I,P\neq 0} \frac{S\cdot R}{Q}$ since $(\Phi, b);(\Phi, b \le J) \vDash Q = \sum_{a\le I} S$ and by definition of H, $(\phi, b, a);(\Phi, b \le J, Q \neq 0, a \le I, S \neq 0) \vDash R\{H/a\} \le R$. Thus, we have $(\phi, b, a);(\Phi, b \le J, Q \neq 0, a \le I) \vDash S \cdot R\{H/a\} \le S \cdot R$. So, in conclusion, we obtain $(\phi, b);(\Phi, b \le J, Q \neq 0);\cdot \mid \cdot \vdash_{\sum_{a\le I,P\neq 0} \frac{S\cdot R}{Q}} v : \tau$. By induction hypothesis, we obtain $\phi;\Phi;\Delta \mid \cdot \vdash_{R_0+\sum_{b\le J,a\le I,P\neq 0,Q\neq 0} S\cdot R} t[y := v] : \nu$. Furthermore, we have $\phi;\Phi \vDash R_0 \le R_1$ and $\phi;\Phi \vDash \sum_{b\le J,a\le I,Q\neq 0,P\neq 0} S \cdot R = \sum_{a\le I,P\neq 0} P \cdot R$. So with a subtyping rule, we obtain $\phi;\Phi;\Gamma \mid \cdot \vdash_{R_1+\sum_{a\le I,P\neq 0} P\cdot R} t[y := v] : \mu$, and this concludes this case.

- Let : Suppose we have the typing

$$\frac{\phi;\Phi;\Gamma \mid \Theta \vdash_{S'} t : \{P : \tau \mid a \le I\} \qquad (\phi, a);(\Phi, a \le I, P \neq 0);\Delta, x : \tau \mid y : \nu \vdash_S u : \mu}{\phi;\Phi;\Gamma,\Delta \mid \Theta + (y : \{P : \nu \mid a \le I\}) \vdash_{1+S'+\sum_{a\le I,P\neq 0} P\cdot S} \texttt{let}\ x = t\ \texttt{in}\ u : \{P : \mu \mid a \le I\}}$$

We consider $\Theta = \emptyset$. Let us note $\nu = \{Q : \sigma \mid b \le J\}$. By definition of convolution, suppose given a proof $(\phi, c);(\Phi, c \le \sum_{a\le I}(J+1), (P\cdot Q)\{\pi_1^a(I,J,c)/a\}\{\pi_2^a(I,J,c)/b\} \neq 0);\cdot \mid \cdot \vdash_R v : \sigma\{\pi_1^a(I,J,c)/a\}\{\pi_2^a(I,J,c)/b\}$. By weakening and substitution we obtain $(\phi, a', b');(\Phi, a' \star^a_{I,J} b' \le \sum_{a\le I}(J+1), a' \le I, b' \le J\{a'/a\}, (P\cdot Q)\{a'/a\}\{b'/b\} \neq 0);\cdot \mid \cdot \vdash_{R\{a'\star^a_{I,J} b'/c\}} v : \sigma\{a'/a\}\{b'/b\}$. Let us call $J' = J\{a'/a\}$. By renaming and contraction: $(\phi, a, b);(\Phi, a \le I, b \le J, P \neq 0, Q \neq 0);\cdot \mid \cdot \vdash_{R\{a\star^{a'}_{I,J'} b/c\}} v : \sigma$. By induction hypothesis, we obtain $(\phi, a);(\Phi, a \le I, P \neq 0);\Delta, x : \tau \mid \cdot \vdash_{S+\sum_{b\le J,Q\neq 0} Q\cdot R\{a\star^{a'}_{I,J'} b/c\}} u[y := v] : \mu$. Then, using the rule for `let`, we obtain the proof $\phi;\Phi;\Gamma,\Delta \mid \cdot \vdash_{1+S'+\sum_{a\le I,P\neq 0} P(S+\sum_{b\le J,Q\neq 0} Q\cdot R\{a\star^{a'}_{I,J'} b/c\})} (\texttt{let}\ x = t\ \texttt{in}\ u)[y := v] : \{P : \mu \mid a \le I\}$. Furthermore,

$\phi;\Phi \vDash \sum_{a\le I,(P\cdot Q)\neq 0, b\le J} P\cdot Q\cdot R\{a\star^{a'}_{I,J'} b/c\} = \sum_{c\le\sum_{a\le I}(J+1),(P\cdot Q)\{\pi^a_1(I,J,c)/a\}\{\pi^a_2(I,J,c)/b\}\neq 0} R\cdot (P\cdot Q)\{\pi^a_1(I,J,c)/a\}\{\pi^a_2(I,J,c)/b\}$, and this concludes the proof for this case. The case $\Theta\neq\emptyset$ is similar.

3.3 Generation Lemma and Subject Reduction

Now, in order to give the subject reduction, we first need to consider the difficult case of values with an arrow type. Indeed, with the rule for introduction and elimination of quantification, a typing derivation for a value with an arrow type is not always a subtyping rule below a syntax-directed rule, such as the λ-abstraction rule, the fixpoint rule or the rule for `Unif`. Trying to explain what this possible chain of subtyping, introduction of quantification and elimination of quantification does is the goal of the following generation lemma.

Lemma 3.11 (Generation Lemma). *Suppose given a proof*

$$\dfrac{\dfrac{\pi}{\phi;\Phi;\cdot\mid\cdot\vdash_R v:\sigma\multimap\mu}\qquad \phi;\Phi;\cdot\mid\cdot\vdash_{R'} w:\sigma}{\phi;\Phi;\cdot\mid\cdot\vdash_{R+R'} v\ w:\mu}$$

Then in the canonical representation of π, that is to say π in which before each non-subtyping rule we have exactly one subtyping rule, for any $(\phi,\overline{b});(\Phi,\overline{\Phi}_b);\cdot\mid\cdot\vdash_{R_0} v:\forall\overline{a}:\overline{\Phi}_a.\sigma_0\multimap\mu_0$, there exists an instantiation $\overline{I}_b,\overline{I}_a$ of $\overline{b},\overline{a}$ valid under $\phi;\Phi$ such that $\phi;\Phi\vDash\overline{\Phi}_b\{\overline{I}_b/\overline{b}\}$, $\phi;\Phi\vDash\overline{\Phi}_a\{\overline{I}_b,\overline{I}_a/\overline{b},\overline{a}\}$, $\phi;\Phi\vdash\mu_0\{\overline{I}_a,\overline{I}_b/\overline{a},\overline{b}\}\sqsubseteq\mu$ and we have a proof $\phi;\Phi;\cdot\mid\cdot\vdash_{R'} w:\sigma_0\{\overline{I}_a,\overline{I}_b/\overline{a},\overline{b}\}$. Furthermore, $\phi;\Phi\vDash R_0\{\overline{I}_b/\overline{b}\}\le R$.

Proof. We prove this lemma by induction, from bottom to top, on the canonical representation of π. We first show that any subproof in the canonical representation of π before the syntax-directed rule has indeed the form $(\phi,\phi');(\Phi,\Phi');\cdot\mid\cdot\vdash_{R_0} v:\sigma_{-\circ}$ for some $\phi',\Phi',\sigma_{-\circ}$. Let us ignore the index R_0 for this first proposition as it does not matter here.

- Base case : The base case, as the induction goes from bottom to top, is $\phi;\Phi;\cdot\mid\cdot\vdash v:\sigma\multimap\mu$ given in the previous typing. This has indeed to good form.
- Elimination of quantification. Let suppose that the bottom proof has indeed the form $\dfrac{}{(\phi,\phi');(\Phi,\Phi');\cdot\mid\cdot\vdash_{R_0} v:\sigma_{-\circ}}$. Thus we have the typing:

$$\dfrac{\dfrac{(\phi,\phi');(\Phi,\Phi');\cdot\mid\cdot\vdash v:\forall a:\Phi_a.\tau_{-\circ}\qquad (\phi,\phi');(\Phi,\Phi')\vDash\Phi_a\{I/a\}}{(\phi,\phi');(\Phi,\Phi');\cdot\mid\cdot\vdash v:\tau_{-\circ}\{I/a\}}\qquad (\phi,\phi');(\Phi,\Phi')\vdash\tau_{-\circ}\{I/a\}\sqsubseteq\sigma_{-\circ}}{(\phi,\phi');(\Phi,\Phi');\cdot\mid\cdot\vdash v:\sigma_{-\circ}}$$

  So we have indeed only typing with the form $(\phi,\phi');(\Phi,\Phi');\cdot\mid\cdot\vdash v:\sigma_{-\circ}$ for some $\phi',\Phi',\sigma_{-\circ}$.
- Other cases : Introduction of quantification works in the same way: it only add new variables and constraints so it does not contradict the form $(\phi,\phi');(\Phi,\Phi');\cdot\mid\cdot\vdash_v:\sigma_{-\circ}$. The other cases are the syntax directed-rule. Note that for value with arrow types, there is exactly 3 possibilities : λ- abstraction, fixpoint and `Unif`, and again those this does not contradict the form given in the theorem.

Now that we know exactly the shape of those proofs. We can prove the proposition.

- Base case : The base case is direct, for $\phi;\Phi;\cdot\mid\cdot\vdash_R v:\sigma\multimap\mu,\overline{a}=\overline{b}=\emptyset$ thus the instantiation is the null instantiation, and we have indeed $\phi;\Phi\vdash\mu\sqsubseteq\mu$ and a proof $\phi,\Phi;\cdot\mid\cdot\vdash_{R'} w:\sigma$. Furthermore, $\phi,\Phi\vDash R\le R$.
- Elimination of quantification : Suppose we have the typing

$$\frac{(\phi,\overline{b});(\Phi,\overline{\Phi}_b);\cdot \mid \cdot \vdash_{R_0} v : \forall c_0 : \Phi_0.(\forall \overline{c} : \overline{\Phi}_c.\tau_0 \multimap \nu_0) \qquad (\phi,\overline{b});(\Phi,\overline{\Phi}_b) \vDash \Phi_0\{I/c_0\}}{(\phi,\overline{b});(\Phi,\overline{\Phi}_b);\cdot \mid \cdot \vdash_{R_0} v : (\forall \overline{c} : \overline{\Phi}_c.\tau_0 \multimap \nu_0)\{I/c_0\}}$$

$$\frac{(\phi,\overline{b});(\Phi,\overline{\Phi}_b) \vdash (\forall \overline{c} : \overline{\Phi}_c.\tau_0 \multimap \nu_0)\{I/c_0\} \sqsubseteq \forall \overline{a} : \overline{\Phi}_a.\sigma_0 \multimap \mu_0}{(\phi,\overline{b});(\Phi,\overline{\Phi}_b);\cdot \mid \cdot \vdash_{R_1} v : \forall \overline{a} : \overline{\Phi}_a.\sigma_0 \multimap \mu_0}$$

With $(\phi,\overline{b});(\Phi,\overline{\Phi}_b) \vDash R_0 \leq R_1$. By induction hypothesis, there exists an instantiation $\overline{I}_b, \overline{I}_a$ of $\overline{b},\overline{a}$ such that $\phi;\Phi \vDash \Phi_b\{\overline{I}_b/\overline{b}\}$, $\phi;\Phi \vDash \Phi_a\{\overline{I}_a,\overline{I}_b/\overline{a},\overline{b}\}$, $\phi;\Phi \vdash \mu_0\{\overline{I}_a,\overline{I}_b/\overline{a},\overline{b}\} \sqsubseteq \mu$ and we have a proof $\phi;\Phi;\cdot \mid \cdot \vdash_{R'} w : \sigma_0\{\overline{I}_a,\overline{I}_b/\overline{a},\overline{b}\}$. Furthermore, $\phi;\Phi \vDash R_1\{\overline{I}_b/\overline{b}\} \leq R$. By definition of subtyping, if we denote $\texttt{elements}(\sigma_0) = \overline{d};\overline{\Phi}_d,\sigma_1$, then there exists $\overline{I}_c$ such that
$(\phi,\overline{b},\overline{a},\overline{d});(\Phi,\overline{\Phi}_b,\overline{\Phi}_a,\overline{\Phi}_d) \vDash \overline{\Phi}_c\{I/c_0\}\{\overline{I}_c/\overline{c}\}$
$(\phi,\overline{b},\overline{a},\overline{d});(\Phi,\overline{\Phi}_b,\overline{\Phi}_a,\overline{\Phi}_d) \vdash \sigma_1 \sqsubseteq \tau_0\{I/c_0\}\{\overline{I}_c/\overline{c}\}$
$(\phi,\overline{b},\overline{a},\overline{d});(\Phi,\overline{\Phi}_b,\overline{\Phi}_a,\overline{\Phi}_d) \vdash \nu_0\{I/c_0\}\{\overline{I}_c/\overline{c}\} \sqsubseteq \mu_0$
Thus, by index substitution, contraction, and definition of `elements`, we have
$\texttt{elements}(\sigma_0\{\overline{I}_a,\overline{I}_b/\overline{a},\overline{b}\}) = \overline{d};\overline{\Phi}_d\{\overline{I}_a,\overline{I}_b/\overline{a},\overline{b}\};\sigma_1\{\overline{I}_a,\overline{I}_b/\overline{a},\overline{b}\}$
$(\phi,\overline{d});(\Phi,\overline{\Phi}_d) \vDash \overline{\Phi}_c\{\overline{I}_b/\overline{b}\}\{I\{\overline{I}_b/\overline{b}\}/c_0\}\{\overline{I}_c\{\overline{I}_a,\overline{I}_b/\overline{a},\overline{b}\}/\overline{c}\}$
$(\phi,\overline{d});(\Phi,\overline{\Phi}_d) \vdash \sigma_1\{\overline{I}_a,\overline{I}_b/\overline{a},\overline{b}\} \sqsubseteq \tau_0\{\overline{I}_b/\overline{b}\}\{I\{\overline{I}_b/\overline{b}\}/c_0\}\{\overline{I}_c\{\overline{I}_a,\overline{I}_b/\overline{a},\overline{b}\}/\overline{c}\}$
$(\phi,\overline{d});(\Phi,\overline{\Phi}_d) \vdash \nu_0\{\overline{I}_b/\overline{b}\}\{I\{\overline{I}_b/\overline{b}\}/c_0\}\{\overline{I}_c\{\overline{I}_a,\overline{I}_b/\overline{a},\overline{b}\}/\overline{c}\} \sqsubseteq \mu_0\{\overline{I}_a,\overline{I}_b/\overline{a},\overline{b}\}$
By Lemma 3.7, there exists an instantiation $\overline{I}_d$ of $\overline{d}$ such that $\phi;\Phi \vDash \overline{\Phi}_d\{\overline{I}_a,\overline{I}_b/\overline{a},\overline{b}\}\{\overline{I}_d/\overline{d}\}$ and we have a proof $\phi;\Phi;\cdot \mid \cdot \vdash_{R'} w : \sigma_1\{\overline{I}_a,\overline{I}_b/\overline{a},\overline{b}\}\{\overline{I}_d/\overline{d}\}$. Let us give the following notation $\overline{J}_c = \overline{I}_c\{\overline{I}_a,\overline{I}_b,\overline{I}_d/\overline{a},\overline{b},\overline{d}\}$ and $J = I\{\overline{I}_b/\overline{b}\}$. Again by index substitution and contraction :
$\phi;\Phi \vDash \Phi_0\{\overline{I}_b/\overline{b}\}\{J/c_0\}$
$\phi;\Phi \vDash \overline{\Phi}_c\{\overline{I}_b/\overline{b}\}\{J/c_0\}\{\overline{J}_c/\overline{c}\}$
$\phi;\Phi \vdash \sigma_1\{\overline{I}_a,\overline{I}_b/\overline{a},\overline{b}\}\{\overline{I}_d/\overline{d}\} \sqsubseteq \tau_0\{\overline{I}_b/\overline{b}\}\{J/c_0\}\{\overline{J}_c/\overline{c}\}$
$\phi;\Phi \vdash \nu_0\{\overline{I}_b/\overline{b}\}\{J/c_0\}\{\overline{J}_c/\overline{c}\} \sqsubseteq \mu_0\{\overline{I}_a,\overline{I}_b/\overline{a},\overline{b}\} \sqsubseteq \mu$
So, the instantiation of $\overline{b},c_0,\overline{c}$ is given by $\overline{I}_b, J, \overline{J}_c$, then $\phi;\Phi \vdash \nu_0\{\overline{I}_b/\overline{b}\}\{J/c_0\}\{\overline{J}_c/\overline{c}\} \sqsubseteq \mu$ is given by transitivity of subtyping, and the proof $\phi;\Phi;\cdot \mid \cdot \vdash_{R'} w : \tau_0\{\overline{I}_b/\overline{b}\}\{J/c_0\}\{\overline{J}_c/\overline{c}\}$ is given by subtyping. Finally, $\phi,\Phi \vDash R_0\{\overline{I}_b/\overline{b}\} \leq R$ by index substitution and transitivity of $\leq$. This concludes this case.

- Introduction of quantification : Suppose we have the typing

$$\frac{(\phi,\overline{b},c_0);(\Phi,\overline{\Phi}_b,\Phi_0);\cdot \mid \cdot \vdash_{R_0} v : \forall \overline{c} : \overline{\Phi}_c.\tau_0 \multimap \nu_0}{(\phi,\overline{b});(\Phi,\overline{\Phi}_b);\cdot \mid \cdot \vdash_{max_{c_0:\Phi_0}(R_0)} v : \forall c_0 : \Phi_0.(\forall \overline{c} : \overline{\Phi}_c.\tau_0 \multimap \nu_0)}$$

$$\frac{(\phi,\overline{b});(\Phi,\overline{\Phi}_b) \vdash \forall c_0 : \Phi_0.(\forall \overline{c} : \overline{\Phi}_c.\tau_0 \multimap \nu_0) \sqsubseteq \forall \overline{a} : \overline{\Phi}_a.\sigma_0 \multimap \mu_0}{(\phi,\overline{b});(\Phi,\overline{\Phi}_b);\cdot \mid \cdot \vdash_{R_1} v : \forall \overline{a} : \overline{\Phi}_a.\sigma_0 \multimap \mu_0}$$

With $(\phi,\overline{b});(\Phi,\overline{\Phi}_b) \vDash max_{c_0:\Phi_0}(R_0) \leq R_1$. By induction hypothesis, there exists an instantiation $\overline{I}_b, \overline{I}_a$ of $\overline{b},\overline{a}$ such that $\phi;\Phi \vDash \Phi_b\{\overline{I}_b/\overline{b}\}$, $\phi;\Phi \vDash \Phi_a\{\overline{I}_a,\overline{I}_b/\overline{a},\overline{b}\}$, $\phi;\Phi \vdash \mu_0\{\overline{I}_a,\overline{I}_b/\overline{a},\overline{b}\} \sqsubseteq \mu$ and we have a proof $\phi;\Phi;\cdot \mid \cdot \vdash_{R'} w : \sigma_0\{\overline{I}_a,\overline{I}_b/\overline{a},\overline{b}\}$. Furthermore, $\phi;\Phi \vDash R_1\{\overline{I}_b/\overline{b}\} \leq R$. By definition of subtyping, if we denote $\texttt{elements}(\sigma_0) = \overline{d};\overline{\Phi}_d,\sigma_1$, then there exists $I,\overline{I}_c$ such that
$(\phi,\overline{b},\overline{a},\overline{d});(\Phi,\overline{\Phi}_b,\overline{\Phi}_a,\overline{\Phi}_d) \vDash \Phi_0\{I/c_0\}$
$(\phi,\overline{b},\overline{a},\overline{d});(\Phi,\overline{\Phi}_b,\overline{\Phi}_a,\overline{\Phi}_d) \vDash \overline{\Phi}_c\{I/c_0\}\{\overline{I}_c/\overline{c}\}$
$(\phi,\overline{b},\overline{a},\overline{d});(\Phi,\overline{\Phi}_b,\overline{\Phi}_a,\overline{\Phi}_d) \vdash \sigma_1 \sqsubseteq \tau_0\{I/c_0\}\{\overline{I}_c/\overline{c}\}$
$(\phi,\overline{b},\overline{a},\overline{d});(\Phi,\overline{\Phi}_b,\overline{\Phi}_a,\overline{\Phi}_d) \vdash \nu_0\{I/c_0\}\{\overline{I}_c/\overline{c}\} \sqsubseteq \mu_0$
As previously, by Lemma 3.7,there exists an instantiation $\overline{I}_d$ of $\overline{d}$ such that $\phi;\Phi \vDash \overline{\Phi}_d\{\overline{I}_a,\overline{I}_b/\overline{a},\overline{b}\}\{\overline{I}_d/\overline{d}\}$

and we have a proof $\phi;\Phi;\cdot \mid \cdot \vdash_{R'} w : \sigma_1\{\overline{I}_a,\overline{I}_b/\overline{a},\overline{b}\}\{\overline{I}_d/\overline{d}\}$. Let us give the following notation $\overline{J}_c = \overline{I}_c\{\overline{I}_a,\overline{I}_b,\overline{I}_d/\overline{a},\overline{b},\overline{d}\}$ and $J = I\{\overline{I}_a,\overline{I}_b,\overline{I}_d/\overline{a},\overline{b},\overline{d}\}$. We obtain :
$\phi;\Phi \vDash \Phi_0\{\overline{I}_b/\overline{b}\}\{J/c_0\}$
$\phi;\Phi \vDash \overline{\Phi}_c\{\overline{I}_b/\overline{b}\}\{J/c_0\}\{\overline{J}_c/\overline{c}\}$
$\phi;\Phi \vdash \sigma_1\{\overline{I}_a,\overline{I}_b/\overline{a},\overline{b}\}\{\overline{I}_d/\overline{d}\} \sqsubseteq \tau_0\{\overline{I}_b/\overline{b}\}\{J/c_0\}\{\overline{J}_c/\overline{c}\}$
$\phi;\Phi \vdash \nu_0\{\overline{I}_b/\overline{b}\}\{J/c_0\}\{\overline{J}_c/\overline{c}\} \sqsubseteq \mu_0\{\overline{I}_a,\overline{I}_b/\overline{a},\overline{b}\} \sqsubseteq \mu$
And we can concludes in the same way as the previous case. The only difference, is for the weight. We have $(\phi,\overline{b});(\Phi,\overline{\Phi}_b) \vDash max_{c_0:\Phi_0}(R_0) \leq R_1$, and so, we have
$\phi;\Phi \vDash max_{c_0:\Phi_0\{\overline{I}_b/\overline{b}\}}(R_0\{\overline{I}_b/\overline{b}\}) \leq R_1\{\overline{I}_b/\overline{b}\} \leq R$ And by definition of maximum, we obtain
$\phi;\Phi \vDash R_0\{\overline{I}_b/\overline{b}\}\{J/c_0\} \leq max_{c_0:\Phi_0\{\overline{I}_b/\overline{b}\}}(R_0\{\overline{I}_b/\overline{b}\}) \leq R_1\{\overline{I}_b/\overline{b}\} \leq R$, and this concludes this case.

- Other cases : The other cases are for the syntax-directed rule, for example λ-abstraction. In this case, the typing would be :

$$\dfrac{\dfrac{(\phi,\overline{b});(\Phi,\overline{\Phi}_b);x:\tau_0 \mid \cdot \vdash_{R_0} t : \nu_0}{(\phi,\overline{b});(\Phi,\overline{\Phi}_b);\cdot \mid \cdot \vdash_{R_0+1} \lambda x.t : \tau_0 \multimap \nu_0} \qquad (\phi,\overline{b});(\Phi,\overline{\Phi}_b) \vdash \tau_0 \multimap \nu_0 \sqsubseteq \forall \overline{a} : \overline{\Phi}_a.\sigma_0 \multimap \mu_0}{(\phi,\overline{b});(\Phi,\overline{\Phi}_b);\cdot \mid \cdot \vdash_{R_1} v : \forall \overline{a} : \overline{\Phi}_a.\sigma_0 \multimap \mu_0}$$

  And so we only need to prove our propriety for the subtyping rule. And this works in the same way as the previous cases, in which we considered subtyping and another rule. This is the same idea for all other syntax-directed rule.

This concludes the proof for the generation lemma. We can now give the main theorem of subject reduction.

Theorem 3.12 (Subject Reduction). *If $\phi;\Phi;\cdot \mid \cdot \vdash_R t : \{P : \sigma \mid a \leq I\}$ and $t \to \{p_i : t_i \mid i \in \mathcal{I}\}$ (with $p_i > 0$ for all i), then for all $i \in \mathcal{I}$, there exists valid rational indexes P_i and R_i such that $\phi;\Phi;\cdot \mid \cdot \vdash_{R_i} t_i : \mu_i$ with $\phi;\Phi \vdash \mu_i \sqsubseteq \{P_i : \sigma \mid a \leq I\}$ and $(\phi,a);(\Phi,a \leq I) \vDash P = \sum_{i\in\mathcal{I}} p_i \cdot P_i$ and $\phi;\Phi \vDash R \geq 1 + \sum_{i\in\mathcal{I}} p_i \cdot R_i$.*

Proof. We prove this by induction on the reduction $t \to \{p_i : t_i \mid i \in \mathcal{I}\}$. First, we show that it is sufficient to prove the subject reduction for a typing derivation for t that does not start with a subtyping rule. Indeed, suppose the subject reduction proved for such typing derivations, and suppose $\phi;\Phi;\cdot \mid \cdot \vdash_R t : \{P : \sigma \mid a \leq I\}$ with $t \to \{p_i : t_i \mid i \in \mathcal{I}\}$. Suppose the typing derivation has the form:

$$\dfrac{\phi;\Phi \vdash R' \leq R \qquad \phi;\Phi;\cdot \mid \cdot \vdash_{R'} t : \{P' : \sigma' \mid a' \leq I'\} \qquad \phi;\Phi \vdash \{P' : \sigma' \mid a' \leq I'\} \sqsubseteq \{P : \sigma \mid a \leq I\}}{\phi;\Phi;\cdot \mid \cdot \vdash_R t : \{P : \sigma \mid a \leq I\}}$$

By subject reduction, we have for all $i \in \mathcal{I}$, $\phi;\Phi;\cdot \mid \cdot \vdash_{R'_i} t : \mu$ with $\phi;\Phi \vdash \mu \sqsubseteq \{P'_i : \sigma' \mid a' \leq I'\}$ and $(\phi,a');(\Phi,a' \leq I') \vDash P' = \sum_{i\in\mathcal{I}} p_i \cdot P'_i$ and $\phi;\Phi \vDash R' \geq 1 + \sum_{i\in\mathcal{I}} p_i \cdot R'_i$. Let us call S the rational index in the coupling given in the proof. We define the rational index S_i by
$S_i = if(P' = 0)\ then\ 0\ else\ \frac{S \cdot P'_i}{P'}$ and we define $P_i = \sum_{a' \leq I'} S_i$. We have then easily
$\phi;\Phi \vdash S_i \lhd_{\sqsubseteq} \langle\{P'_i : \sigma' \mid a' \leq I'\}\&\{P_i : \sigma \mid a \leq I\}\rangle$. Thus,
$\phi;\Phi \vdash \mu \sqsubseteq \{P'_i : \sigma' \mid a' \leq I'\} \sqsubseteq \{P_i : \sigma \mid a \leq I\}$. Moreover
$(\phi,a);(\Phi,a \leq I) \vDash \sum_{i\in\mathcal{I}} p_i \cdot P_i = P$ using the fact that $(\phi,a,a');(\Phi,a \leq I,a' \leq I') \vDash \sum_{i\in\mathcal{I}} p_i \cdot S_i = S$. Moreover, $\phi;\Phi \vDash R \geq R' \geq 1 + \sum_{i\in\mathcal{I}} p_i \cdot R'_i$. This concludes the subject reduction in the case of a subtyping rule. Thus, in the typing $\phi;\Phi;\cdot \mid \cdot \vdash_R t : \{P : \sigma \mid a \leq I\}$, we can now suppose that the first rule is not a subtyping rule.

- Beta reduction : If the term is $(\lambda x.t)\ v$, then we have the typing :

$$\dfrac{\dfrac{\pi}{\phi;\Phi;\cdot \mid \cdot \vdash_R \lambda x.t : \sigma \multimap \mu} \qquad \phi;\Phi;\cdot \mid \cdot \vdash_{R'} v : \sigma}{\phi;\Phi;\cdot \mid \cdot \vdash_{R+R'} (\lambda x.t)\, v : \mu}$$

Then, by lemma 3.11, for this subproof in π :

$$\dfrac{(\phi,\overline{b});(\Phi,\overline{\Phi}_b);x:\sigma_0 \mid \cdot \vdash_{R_0} t : \mu_0}{(\phi,\overline{b});(\Phi,\overline{\Phi}_b);\cdot \mid \cdot \vdash_{R_0+1} \lambda x.t : \sigma_0 \multimap \mu_0}$$

There exists an instantiation $\overline{I}_b$ of $\overline{b}$ such that $\phi;\Phi \vDash \overline{\Phi}_b\{\overline{I}_b/\overline{b}\}$ and $\phi;\Phi \vdash \mu_0\{\overline{I}_b/\overline{b}\} \sqsubseteq \mu$ and we have a proof $\phi;\Phi;\cdot \mid \cdot \vdash_{R'} v : \sigma_0\{\overline{I}_b/\overline{b}\}$. Furthermore, $\phi;\Phi \vDash (R_0+1)\{\overline{I}_b/\overline{b}\} \leq R$. By Lemma 3.3 and 3.2, we have a proof $\phi;\Phi;x:\sigma_0\{\overline{I}_b/\overline{b}\} \mid \cdot \vdash_{R_0\{\overline{I}_b/\overline{b}\}} t : \mu_0\{\overline{I}_b/\overline{b}\}$. Then, by Lemma 3.8, we obtain a proof $\phi;\Phi;\cdot \mid \cdot \vdash_{R'+R_0\{\overline{I}_b/\overline{b}\}} t[x := v] : \mu_0\{\overline{I}_b/\overline{b}\}$.And we have $\phi;\Phi \vDash R + R' \geq 1 + R' + R_0\{\overline{I}_b/\overline{b}\}$. This concludes this case.

- Tensor : The case for tensor is another application of Lemma 3.8.
- Unif : If the term is $\texttt{Unif}\ \underline{n}$, then we have the typing :

$$\dfrac{\dfrac{\pi}{\phi;\Phi;\cdot \mid \cdot \vdash_R \texttt{Unif} : \sigma \multimap \mu} \qquad \phi;\Phi;\cdot \mid \cdot \vdash_{R'} \underline{n} : \sigma}{\phi;\Phi;\cdot \mid \cdot \vdash_{R+R'} \texttt{Unif}\ \underline{n} : \mu = \{Q : \tau \mid b \leq J\}}$$

So, by lemma 3.11, for this subproof in π :

$$\dfrac{}{(\phi,\overline{b});(\Phi,\overline{\Phi}_b);\cdot \mid \cdot \vdash_1 \texttt{Unif} : \mathsf{Nat}(I) \multimap \{\frac{1}{I+1} : \mathsf{Nat}(a) \mid a \leq I\}}$$

There exists an instantiation $\overline{I}_b$ of $\overline{b}$ such that $\phi;\Phi \vDash \overline{\Phi}_b\{\overline{I}_b/\overline{b}\}$ and $\phi;\Phi \vdash \{\frac{1}{I\{\overline{I}_b/\overline{b}\}+1} : \mathsf{Nat}(a) \mid a \leq I\{\overline{I}_b/\overline{b}\}\} \sqsubseteq \mu$ and we have a proof $\phi;\Phi;\cdot \mid \cdot \vdash_{R'} \underline{n} : \mathsf{Nat}(I\{\overline{I}_b/\overline{b}\})$. Furthermore, $\phi;\Phi \vDash 1 \leq R$. By Lemma 3.5, we obtain $\phi;\Phi \vDash n = I\{\overline{I}_b/\overline{b}\}$. We can then take the simpler hypothesis:
$\phi;\Phi \vdash \{\frac{1}{n+1} : \mathsf{Nat}(a) \mid a \leq n\} \sqsubseteq \mu$. Recall that the reduction gives us
$\texttt{Unif}\ \underline{n} \longrightarrow \{\frac{1}{n+1} : \underline{m} \mid m \leq n\}$. Let m be an integer such that $m \leq n$. We have the typing $\phi;\Phi;\cdot \mid \cdot \vdash_0 \underline{m} : \mathsf{Nat}(m)$. Let us call S the rational index such that
$\phi;\Phi \vdash S \lhd_{\sqsubseteq} \langle\{\frac{1}{n+1} : \mathsf{Nat}(a) \mid a \leq n\} \& \{Q : \tau \mid b \leq J\}\rangle$. We define the rational index $S_m = S\{m/a\} \cdot (n+1)$. By index substitution and contraction (for a) and weakening (for c), we obtain that S_m is valid under $(\phi,b,c);(\Phi, b \leq J, c \leq 0)$.
Let us define $Q_m = S_m$. We have that Q_m is valid under $(\phi,b);(\Phi, b \leq J)$. Moreover, $(\phi,b);(\Phi, b \leq J) \vDash \sum_{c\leq 0} S_m = Q_m$. We have also, using the coupling S,
$(\phi,c);(\Phi, c \leq 0) \vDash \sum_{b\leq J} S\{m/a\} \cdot (n+1) = 1$. Moreover,
$(\phi,b,c);(\Phi, b \leq J, c \leq 0, S\{m/a\} \neq 0) \vdash \mathsf{Nat}(m) \sqsubseteq \tau$. Finally, all this gives us
$\phi;\Phi \vdash \{1 : \mathsf{Nat}(m) \mid c \leq 0\} \sqsubseteq \{Q_m : \tau \mid b \leq J\}$. And we have
$(\phi,b);(\Phi, b \leq J) \vDash \sum_{m\leq n} \frac{Q_m}{n+1} = \sum_{m\leq n} S\{m/a\} = \sum_{a\leq n} S = Q$. We have also directly $\phi;\Phi \vdash R + R' \geq 1$. This concludes the proof for this case.
- Let value: If the term is $\texttt{let}\ x = v\ \texttt{in}\ t$, we have the typing:

$$\dfrac{\phi;\Phi;\cdot \mid \cdot \vdash_{R'} v : \{1 : \sigma \mid c \leq 0\} \qquad (\phi,c);(\Phi, c \leq 0, 1 \neq 0);x:\sigma \mid \cdot \vdash_R t : \mu}{\phi;\Phi;\cdot \mid \cdot \vdash_{1+R'+R\{1/c\}} \texttt{let}\ x = v\ \texttt{in}\ t : \mu\{1/c\}}$$

By index substitution and contraction, we have a proof $\phi;\Phi;x:\sigma \mid \cdot \vdash_{R\{1/c\}} t : \mu\{1/c\}$ and $\phi;\Phi;\cdot \mid \cdot \vdash_{R'} v : \sigma$. By lemma 3.8, we have $\phi;\Phi;\cdot \mid \cdot \vdash_{R'+R\{1/c\}} t[x := v] : \mu\{1/c\}$. And this concludes the case.
- Let active term: Suppose the term is $\texttt{let}\ x = t\ \texttt{in}\ u$, with t an active term such that $t \longrightarrow \{p_i : t_i \mid i \in \mathcal{I}\}$, so $\texttt{let}\ x = t\ \texttt{in}\ u \longrightarrow \{p_i : \texttt{let}\ x = t_i\ \texttt{in}\ u \mid i \in \mathcal{I}\}$. We have the typing

$$\frac{\phi;\Phi;\cdot \mid \cdot \vdash_{R'} t : \{P : \sigma \mid a \le I\} \qquad (\phi,a);(\Phi, a \le I, P \neq 0); x:\sigma \mid \cdot \vdash_R u : \mu}{\phi;\Phi;\cdot \mid \cdot \vdash_{1+R'+\sum_{a\le I, P\neq 0} P\cdot R} \texttt{let } x = t \texttt{ in } u : \{P : \mu \mid a \le I\}}$$

And by induction hypothesis, for all $i \in \mathcal{I}$ we have a proof $\phi;\Phi;\cdot \mid \cdot \vdash_{R'_i} t_i : v_i$ with $\phi;\Phi \vdash v_i \sqsubseteq \{P_i : \sigma \mid a \le I\}$ and $(\phi,a);(\Phi, a \le I) \vDash \sum_{i\in\mathcal{I}} p_i \cdot P_i = P$ and $\phi;\Phi \vDash 1 + \sum_{i\in\mathcal{I}} p_i \cdot R'_i \le R'$. Note that the first equality show that $(\phi,a);(\Phi, a \le I, P_i \neq 0) \vDash P \neq 0$. With this, using weakening and contraction, we obtain the proof $(\phi,a);(\Phi, a \le I, P_i \neq 0); x:\sigma \mid \cdot \vdash_R u : \mu$. Using the subtyping rule for t, as t is an active term, we can derive the proof $\phi;\Phi;\cdot \mid \cdot \vdash_{R'_i} t_i : \{P_i : \sigma \mid a \le I\}$. Thus, using the `let` rule, we obtain the typing $\phi;\Phi;\cdot \mid \cdot \vdash_{1+R'_i+\sum_{a\le I, P_i\neq 0} P_i\cdot R} \texttt{let } x = t_i \texttt{ in } u : \{P_i : \mu \mid a \le I\}$. It is then easy to see by definition of convolution that the sum over all $i \in I$ of the weight for the DDT of u is equal to the original one in the previous typing. Now we only need to show the inequality $\phi;\Phi \vDash 1 + R' + \sum_{a\le I, P\neq 0} P \cdot R \ge 1 + \sum_{i\in\mathcal{I}} p_i \cdot (1 + R'_i + \sum_{a\le I, P_i\neq 0} P_i \cdot R)$. Again, this is rather direct using the previous hypothesis on R' and P and using the fact that $P_i \neq 0$ implies $P \neq 0$. This concludes this case.

- Match zero: Suppose the term is $\texttt{match z with } \{\ \texttt{z} \mapsto\ t \mid \texttt{s} \mapsto\ w\ \}$. We have the typing :

$$\frac{\begin{array}{cc}\phi;(\Phi,\Phi_a\{0/a\});\cdot \mid \cdot \vdash_R t : \mu & (\phi,a);(\Phi,\Phi_a\{a+1/a\}) \vDash \Phi' \\ \phi;\Phi;\cdot \mid \cdot \vdash_{R'} \texttt{z} : \mathsf{Nat}(a \mid \Phi_a) & \phi;(\Phi,\Phi');\cdot \mid \cdot \vdash_R w : \mathsf{Nat}(a \mid \Phi_a\{a+1/a\}) \multimap \mu\end{array}}{\phi;\Phi;\cdot \mid \cdot \vdash_{1+R'+R} \texttt{match z with } \{\ \texttt{z} \mapsto\ t \mid \texttt{s} \mapsto\ w\ \} : \mu}$$

  By Lemma 3.5, the typing $\phi;\Phi;\cdot \mid \cdot \vdash_{R'} \texttt{z} : \mathsf{Nat}(a \mid \Phi_a)$ gives us $\phi;\Phi \vDash \Phi_a\{0/a\}$. So, by contraction, we have the proof $\phi;\Phi;\cdot \mid \cdot \vdash_R t : \mu$. As $\phi;\Phi \vDash 1 + R' + R \ge 1 + R$, this concludes this case.

- Match successor: Suppose the term is $\texttt{match } s(v) \texttt{ with } \{\ \texttt{z} \mapsto\ t \mid \texttt{s} \mapsto\ w\ \}$. We have the typing :

$$\frac{\begin{array}{cc}\phi;\Phi;\cdot \mid \cdot \vdash_{R'} s(v) : \mathsf{Nat}(a \mid \Phi_a) & (\phi,a);(\Phi,\Phi_a\{a+1/a\}) \vDash \Phi' \\ \phi;(\Phi,\Phi_a\{0/a\});\cdot \mid \cdot \vdash_R t : \mu & \phi;(\Phi,\Phi');\cdot \mid \cdot \vdash_R w : \mathsf{Nat}(a \mid \Phi_a\{a+1/a\}) \multimap \mu\end{array}}{\phi;\Phi;\cdot \mid \cdot \vdash_{1+R'+R} \texttt{match } s(v) \texttt{ with } \{\ \texttt{z} \mapsto\ t \mid \texttt{s} \mapsto\ w\ \} : \mu}$$

  By Lemma 3.5, the typing $\phi;\Phi;\cdot \mid \cdot \vdash_{R'} s(v) : \mathsf{Nat}(a \mid \Phi_a)$ show that $s(v) = \underline{n+1}$ for some integer n, and $\phi;\Phi \vDash \Phi_a\{n+1/a\}$. So, we have $\phi;\Phi \vDash (\Phi_a\{a+1/a\})\{n/a\}$. Again by Lemma 3.5, we have $\phi;\Phi;\cdot \mid \cdot \vdash_0 v = \underline{n} : \mathsf{Nat}(a \mid \Phi_a\{a+1/a\})$. Moreover, by index substitution in the proof $(\phi,a);(\Phi,\Phi_a\{a+1/a\}) \vDash \Phi'$, we obtain $\phi;(\Phi,\Phi_a\{a+1/a\}\{n/a\}) \vDash \Phi'$. Indeed, recall that a is not free in Φ and Φ'. Thus, by contraction we have $\phi;\Phi \vDash \Phi'$. And again by contraction, we obtain the proof $\phi;\Phi;\cdot \mid \cdot \vdash_R w : \mathsf{Nat}(a \mid \Phi_a\{a+1/a\}) \multimap \mu$. Then using the rule for application, we obtain $\phi;\Phi;\cdot \mid \cdot \vdash_R w\ v : \mu$, and this concludes the case.

- Fixpoint: Suppose the term is $(\texttt{fix } x.v)\ w$. We have the proof

$$\frac{\dfrac{\pi}{\phi;\Phi;\cdot \mid \cdot \vdash_R \texttt{fix } x.v : \sigma \multimap \mu} \qquad \phi;\Phi;\cdot \mid \cdot \vdash_{R'} w : \sigma}{\phi;\Phi\cdot \mid \cdot \vdash_{R+R'} (\texttt{fix } x.v)\ w : \mu}$$

  In π, consider the subproof :

$$\frac{\begin{array}{c}(\phi,\overline{b},b);(\Phi,\overline{\Phi}_b) \vDash Q \ge 1 + R_0 + \sum_{a\le I, P\neq 0} P \cdot Q\{M/b\} \\ (\phi,\overline{b},b);(\Phi,\overline{\Phi}_b);\cdot \mid y : \{P : \tau\{M/b\} \mid a \le I\} \vdash_{R_0} v : \tau\end{array}}{(\phi,\overline{b});(\Phi;\overline{\Phi}_b);\cdot \mid \cdot \vdash_{Q\{J/b\}} \texttt{fix } y.v : (\forall \overline{a} : \overline{\Phi}_a.\sigma_0 \multimap \mu_0)\{J/b\} = \tau\{J/b\}}$$

  By Lemma 3.11, there exists an instantiation $\overline{I}_b, \overline{I}_a$ of $\overline{b}, \overline{a}$ valid under $\phi;\Phi$ such that
  $\phi;\Phi \vDash \overline{\Phi}_b\{\overline{I}_b/\overline{b}\}$
  $\phi;\Phi \vDash \overline{\Phi}_a\{J/b\}\{\overline{I}_b, \overline{I}_a/\overline{b}, \overline{a}\}$
  $\phi;\Phi \vdash \mu_0\{J/b\}\{\overline{I}_b, \overline{I}_a/\overline{b}, \overline{a}\} \sqsubseteq \mu$

$\phi;\Phi;\cdot \mid \cdot \vdash_{R'} w : \sigma_0\{J/b\}\{\overline{I}_b, \overline{I}_a/\overline{b}, \overline{a}\}$
$\phi;\Phi \vDash Q\{J/b\}\{\overline{I}_b/\overline{b}\} \leq R$
Then by index substitution and contraction, we have :

$$\frac{(\phi,b);\Phi \vDash Q\{\overline{I}_b/\overline{b}\} \geq 1 + R_0\{\overline{I}_b/\overline{b}\} + \sum_{a \leq I\{\overline{I}_b/\overline{b}\}, P\{\overline{I}_b/\overline{b}\} \neq 0} P\{\overline{I}_b/\overline{b}\} \cdot Q\{\overline{I}_b/\overline{b}\}\{M\{\overline{I}_b/\overline{b}\}/b\} \quad (\phi,b);\Phi;\cdot \mid y : \{P\{\overline{I}_b/\overline{b}\} : \tau\{\overline{I}_b/\overline{b}\}\{M\{\overline{I}_b/\overline{b}\}/b\} \mid a \leq I\{\overline{I}_b/\overline{b}\}\} \vdash_{R_0\{\overline{I}_b/\overline{b}\}} v : \tau\{\overline{I}_b/\overline{b}\}}{\phi;\Phi;\cdot \mid \cdot \vdash_{Q\{\overline{I}_b/\overline{b}\}\{J\{\overline{I}_b/\overline{b}\}/b\}} \texttt{fix}\ y.v : (\forall \overline{a} : \overline{\Phi}_a\{\overline{I}_b/\overline{b}\}.\sigma_0\{\overline{I}_b/\overline{b}\} \multimap \mu_0\{\overline{I}_b/\overline{b}\})\{J\{\overline{I}_b/\overline{b}\}/b\}}$$

By using the notation $A\{\overline{I}_b/\overline{b}\} = A'$, we have :

$$\frac{(\phi,b);\Phi \vDash Q' \geq 1 + R'_0 + \sum_{a \leq I', P' \neq 0} P' \cdot Q'\{M'/b\} \quad (\phi,b);\Phi;\cdot \mid y : \{P' : \tau'\{M'/b\} \mid a \leq I'\} \vdash_{R'_0} v : \tau'}{\phi;\Phi;\cdot \mid \cdot \vdash_{Q'\{J'/b\}} \texttt{fix}\ y.v : (\forall \overline{a} : \overline{\Phi}'_a.\sigma'_0 \multimap \mu'_0)\{J'/b\} = \tau'\{J'/b\}}$$

Remark that the premises for this typing do not use the index J', so the typing is valid even if we change this index. For the sake of simplicity, let us denote $M'_J = M'\{a'/a\}\{J'/b\}$. We also define P'_J in the same way. We also note $\tau'_J = \tau'\{M'_J/b\}$. By weakening, we obtain the proof $(\phi,a');(\Phi,a' \leq I'\{J'/b\}, P'_J \neq 0);\cdot \mid \cdot \vdash_{Q\{M'_J/b\}} \texttt{fix}\ y.v : \tau'_J$. By index substitution and renaming $\phi;\Phi;\cdot \mid y : \{P'_J : \tau'_J \mid a' \leq I'\{J'/b\}\} \vdash_{R'_0\{J'/b\}} v : \tau'\{J'/b\}$. Using Lemma 3.10, we obtain $\phi;\Phi;\cdot \mid \cdot \vdash_{R'_0\{J'/b\} + \sum_{a' \leq I'\{J'/b\}, P'_J \neq 0} P'_J \cdot Q'\{M'_J/b\}} v[y := \texttt{fix}\ y.v] : \tau'\{J'/b\}$. Then, again by index substitution and renaming, we have
$\phi;\Phi \vDash Q'\{J'/b\} \geq 1 + R'_0\{J'/b\} + \sum_{a' \leq I'\{J'/b\}, P'_J \neq 0} P'_J \cdot Q'\{M'_J/b\}$. Recall that we know:
$\phi;\Phi \vDash \overline{\Phi}'_a\{J'/b\}\{\overline{I}_a/\overline{a}\}$
$\phi;\Phi \vdash \mu'_0\{J'/b\}\{\overline{I}_a/\overline{a}\} \sqsubseteq \mu$
$\phi;\Phi;\cdot \mid \cdot \vdash_{R'} w : \sigma'_0\{J'/b\}\{\overline{I}_a/\overline{a}\}$
$\phi;\Phi \vDash Q'\{J'/b\} \leq R$
Let us call $S = R'_0\{J'/b\} + \sum_{a' \leq I'\{J'/b\}, P'_J \neq 0} P'_J \cdot Q'\{M'_J/b\}$. We can construct the proof

$$\frac{\phi;\Phi;\cdot \mid \cdot \vdash_S v[y := \texttt{fix}\ y.v] : \forall \overline{a} : \overline{\Phi}'_a\{J'/b\}.\sigma'_0\{J'/b\} \multimap \mu'_0\{J'/b\} \qquad \phi;\Phi \vDash \overline{\Phi}'_a\{J'/b\}\{\overline{I}_a/\overline{a}\}}{\phi;\Phi;\cdot \mid \cdot \vdash_S v[y := \texttt{fix}\ y.v] : \sigma'_0\{J'/b\}\{\overline{I}_a/\overline{a}\} \multimap \mu'_0\{J'/b\}\{\overline{I}_a/\overline{a}\}}$$

By using several instances of the elimination of quantification rule. Then, using the application rule, we obtain a proof $\phi;\Phi;\cdot \mid \cdot \vdash_{S+R'} (v[y := \texttt{fix}\ y.v])\ w : \mu'_0\{J'/b\}\{\overline{I}_a/\overline{a}\}$, and we have indeed $\phi;\Phi \vDash R + R' \geq Q'\{J'/b\} + R' \geq 1 + S + R'$. This concludes the proof for this case.

3.4 Multi-Distributions, Reduction Rules and Subject Reduction

Now we would like to give the subject reduction theorem in a form that consider the relation on distribution $\Rightarrow$ and show the link with expected runtime. However, we can show informally on a simple example that this is not as simple as it seems. Indeed, take the term $v \oplus_1 v$. This term will obviously reduce to $\{1 : v\}$. However, we could have two distinct typing derivation $\phi;\Phi;\cdot \mid \cdot \vdash_R v : \sigma$ and $\phi;\Phi;\cdot \mid \cdot \vdash_{R'} v : \sigma'$ and the typing derivation $\phi;\Phi;\cdot \mid \cdot \vdash_{\frac{R+R'}{2}} v \oplus_1 v : \{\frac{1}{2} : \sigma; \frac{1}{2} : \sigma'\}$, and then the reduction rule does not take into account this multiplicity of typing derivation. The same problem occurs in [Dal Lago and Grellois 2017]. To take this multiplicity in account, we modify the reduction rule $\Rightarrow$ on distribution to a reduction rule on multi distribution $\Rightarrow_M$, and for example, we have $v \oplus_1 v \Rightarrow_M \{\{\frac{1}{2} : v; \frac{1}{2} : v\}\}$. Let us present formally those notions.

Definition 3.13 (Multi-Distribution). A *multi-distribution* $\mathcal{M}$ on a countable set X is a distribution on a multi-set of X. We can then define the notion of *proper* and *finite* multi-distribution as

previously. We denote such a multi-distribution by $\{\{\mathcal{M}(i) : x_i \mid i \in \mathcal{I}\}\}$ with for all $i \in \mathcal{I}$, $x_i \in X$ and $\sum_{i\in\mathcal{I}} \mathcal{M}(i) \leq 1$.

We can also define as expected the multi-distribution $p \cdot \mathcal{M}$. However, the definition of sum differs from the previous one for distribution. This time, the "sum" of two multi-distributions is the disjoint union that we note $\coprod$. We can also define the decomposition of multi-distributions into active terms and values as expected. We denote this decomposition with $\coprod_{a/v}$. We can now present the distribution represented by a multi-distribution.

Definition 3.14 (Representation). Given a multi distribution $\mathcal{M} = \{\{\mathcal{M}(i) : x_i \mid i \in \mathcal{I}\}\}$ on X, we say that $\mathcal{M}$ is a representation of a distribution $\mathcal{D}$ if $\forall x \in X$, $\mathcal{D}(x) = \sum_{i\in\mathcal{I}, x_i = x} \mathcal{M}(i)$.

Given a distribution $\mathcal{D} = \{p_i : x_i \mid i \in \mathcal{I}\}$, we note $\{\mathcal{D}\}$ the multi-distribution $\{\{p_i : x_i \mid i \in \mathcal{I}\}\}$. Then trivially, $\{\mathcal{D}\}$ is a representation of $\mathcal{D}$. We also give some good proprieties of representation:

Lemma 3.15 (Sum and Representation). *If $\mathcal{M}$ is a representation of $\mathcal{D}$ and $\mathcal{N}$ is a representation of $\mathcal{E}$, then $p \cdot \mathcal{M}$ is a representation of $p \cdot \mathcal{D}$ and $\mathcal{M} \coprod \mathcal{N}$ is a representation of $\mathcal{D} + \mathcal{E}$. Moreover, if $\mathcal{M} = \mathcal{M}_a \coprod_{a/v} \mathcal{M}_v$ and $\mathcal{M}$ is a representation of $\mathcal{D} = \mathcal{D}_a +_{a/v} \mathcal{D}_v$ then $\mathcal{M}_v$ is a representation of $\mathcal{D}_v$ and $\mathcal{M}_a$ is a representation of $\mathcal{D}_a$*

Finally we can give the definition of the reduction rule for multi distribution $\Rightarrow_{MD}$. For this, we keep the previous definition of $\rightarrow$ in Figure 2 and we use this rule :

$$\frac{\mathcal{M} = \{\{p_i : t_i \mid i \in \mathcal{I}\}\} \coprod_{a/v} \mathcal{M}_v \quad \forall i \in \mathcal{I}, t_i \rightarrow \mathcal{E}_i}{\mathcal{M} \Rightarrow_M \mathcal{M}_v \coprod (\coprod_{i\in\mathcal{I}} p_i \cdot \{\mathcal{E}_i\})}$$

With the previous lemma, we obtain easily that the multi-distribution obtained by a reduction step $\Rightarrow_M$ is a representation of the distribution obtained by a reduction step $\Rightarrow$.

Lemma 3.16 (Coherence Between Distributions and Multi-Distributions Reductions). *If $\mathcal{D} \Rightarrow^n \mathcal{E}$ then $\{\mathcal{D}\} \Rightarrow^n_M \mathcal{N}$ with $\mathcal{N}$ a representation of $\mathcal{E}$.*

We can then give the subject reduction according to this new reduction rule.

Theorem 3.17 (Subject Reduction for Multi-Distributions). *Suppose given a proof $\phi; \Phi; \cdot \mid \cdot \vdash_R t : \{P : \sigma \mid a \leq I\}$ and $\{\{1 : t\}\} \Rightarrow^n_M \{\{p_i : t_i \mid i \in \mathcal{I}\}\}$ (with $p_i > 0$ for all i), then for all $i \in \mathcal{I}$, there exists valid rational indexes P_i and R_i such that $\phi; \Phi; \cdot \mid \cdot \vdash_{R_i} t_i : \mu_i$ with $\phi; \Phi \vdash \mu_i \sqsubseteq \{P_i : \sigma \mid a \leq I\}$ and $(\phi, a); (\Phi, a \leq I) \vDash P = \sum_{i\in\mathcal{I}} p_i \cdot P_i$ and $\phi; \Phi \vDash R \geq \sum_{m=0}^{n-1} (\sum \mathcal{D}^m_a) + \sum_{i\in\mathcal{I}} p_i \cdot R_i$ with $\mathcal{D}^m_a$ the distribution such that $t \Rightarrow^m_a \mathcal{D}^m_a$.*

Proof. We prove this by induction on n. The case $n = 0$ is trivial. So we consider the case $n + 1$ for $n \geq 0$. We have a proof $\phi; \Phi; \cdot \mid \cdot \vdash_R t : \{P : \sigma \mid a \leq I\}$ Let us call $\mathcal{M}$ the multi-distribution such that $\{\{1 : t\}\} \Rightarrow^n_M \mathcal{M}$. $\mathcal{M}$ can be decomposed into $\mathcal{M} = \{\{q_j : t_j \mid j \in \mathcal{J}\}\} \coprod_{a/v} \{\{p_i : v_i \mid i \in \mathcal{I}\}\}$. We have $t_j \rightarrow \mathcal{E}_j = \{q_{j,k} : t_{j,k} \mid k \in \mathcal{K}\}$, and so by definition, we have $\{\{1 : t\}\} \Rightarrow^n_M \{\{p_i : v_i \mid i \in \mathcal{I}\}\} \coprod (\coprod_{j\in\mathcal{J}} q_j \cdot \{\{q_{j,k} : t_{j,k} \mid k \in \mathcal{K}_j\}\})$.

By induction hypothesis:

- For all $i \in \mathcal{I}$, we have $\phi; \Phi; \cdot \mid \cdot \vdash_{R_i} v_i : \mu_i$ with $\phi; \Phi \vdash \mu_i \sqsubseteq \{P_i : \sigma \mid a \leq I\}$.
- For all $j \in \mathcal{J}$, we have $\phi; \Phi; \cdot \mid \cdot \vdash_{S_j} t_j : \nu_j$ with $\phi; \Phi \vdash \nu_j \sqsubseteq \{Q_j : \sigma \mid a \leq I\}$.
- $(\phi, a); (\Phi, a \leq I) \vDash P = \sum_{i\in\mathcal{I}} p_i \cdot P_i + \sum_{j\in\mathcal{J}} q_j \cdot Q_j$.
- $\phi; \Phi \vDash R \geq \sum_{m=0}^{n-1} (\sum \mathcal{D}^m_a) + \sum_{i\in\mathcal{I}} p_i \cdot R_i + \sum_{j\in\mathcal{J}} q_j \cdot S_j$.

We can now use Theorem 3.12 on the proof $\phi;\Phi;\cdot \mid \cdot \vdash_{S_j} t_j : \{Q_j : \sigma \mid a \leq I\}$ obtained by subtyping. So, for all $j \in \mathcal{J}$, we have

- For all $k \in \mathcal{K}_j$, we have $\phi;\Phi;\cdot \mid \cdot \vdash_{S_{j,k}} t_{j,k} : v_{j,k}$ with $\phi;\Phi \vdash v_{j,k} \sqsubseteq \{Q_{j,k} : \sigma \mid a \leq I\}$
- $(\phi, a);(\Phi, a \leq I) \vDash Q_j = \sum_{k\in\mathcal{K}_j} q_{j,k} \cdot Q_{j,k}$
- $\phi;\Phi \vDash S_j \geq 1 + \sum_{k\in\mathcal{K}_j} q_{j,k} \cdot S_{j,k}$

So, in conclusion, we obtain :

- For all $i \in \mathcal{I}$, we have $\phi;\Phi;\cdot \mid \cdot \vdash_{R_i} v_i : \mu_i$ with $\phi;\Phi \vdash \mu_i \sqsubseteq \{P_i : \sigma \mid a \leq I\}$
- For all $j \in \mathcal{J}, k \in \mathcal{K}$, we have $\phi;\Phi;\cdot \mid \cdot \vdash_{S_{j,k}} t_{j,k} : v_{j,k}$ with $\phi;\Phi \vdash v_{j,k} \sqsubseteq \{Q_{j,k} : \sigma \mid a \leq I\}$
- $(\phi, a);(\Phi, a \leq I) \vDash P = \sum_{i\in\mathcal{I}} p_i \cdot P_i + \sum_{j\in\mathcal{J},k\in\mathcal{K}_j} q_j \cdot q_{j,k} \cdot Q_{j,k}$.
- $\phi;\Phi \vDash R \geq \sum_{m=0}^{n-1} (\sum \mathcal{D}_a^m) + \sum_{i\in\mathcal{I}} p_i \cdot R_i + \sum_{j\in\mathcal{J}} q_j \cdot S_j$

  $\phi;\Phi \vDash R \geq \sum_{m=0}^{n-1} (\sum \mathcal{D}_a^m) + \sum_{i\in\mathcal{I}} p_i \cdot R_i + \sum_{j\in\mathcal{J}} q_j \cdot (1 + \sum_{k\in\mathcal{K}_j} q_{j,k} \cdot S_{j,k})$

  $\phi;\Phi \vDash R \geq \sum_{m=0}^{n-1} (\sum \mathcal{D}_a^m) + \sum_{i\in\mathcal{I}} p_i \cdot R_i + \sum_{j\in\mathcal{J}} q_j + \sum_{j\in\mathcal{J},k\in\mathcal{K}_j} q_j \cdot q_{j,k} \cdot S_{j,k}$

  $\phi;\Phi \vDash R \geq \sum_{m=0}^{n} (\sum \mathcal{D}_a^m) + \sum_{i\in\mathcal{I}} p_i \cdot R_i + \sum_{j\in\mathcal{J},k\in\mathcal{K}_j} q_j \cdot q_{j,k} \cdot S_{j,k}$

And this concludes this proof.

4 BACK TO OUR EXAMPLE

4.1 Extension of the Type System for Lists

In order to describe our example on probabilistic quicksort, we add to our calculus constructors and types for lists. Those constructors and those types follow the same mechanic as the one for integers. Formally, we add the values `nil` and $\mathsf{cons}(v, w)$. We also add the active term $\mathtt{match}\ v\ \mathtt{with}\ \{\ \mathtt{nil} \mapsto t \mid \mathsf{cons} \mapsto w\}$. Then reduction is defined as expected.

Definition 4.1 (List Indexes). For lists, we introduce a new kind of indexes : *List Indexes*. They are given by the grammar $L ::= l \mid nil \mid cons(I, L) \mid tl(L)$. The variable l denotes a list variable and $tl(L)$ is the tail of the list. Remark that, contrary to natural indexes, we restrict the set of functions on list indexes to the essentials. However, we add new functions for natural indexes, such as $len(L)$ returning the length of a list and $L.(I)$ returning the I^{th} element of L if it exists and 0 otherwise. We will describe other functions when needed.

The, we can describe types for lists in the same way as integers.

Definition 4.2 (Types for Lists). In linear dependent types σ, τ, we add the type $\mathsf{List}(l \mid \Phi_l)$, with l free in Φ_l. This type follows the same intuition as the type $\mathsf{Nat}(a \mid \Phi_a)$ for integers. Again, we write $\mathsf{List}(L)$ the type $\mathsf{List}(l \mid l = L)$ and we define validity as expected. We also add for arrow types the quantification over list variables: $\forall l : \Phi.\sigma_{\_\circ}$. The counterpart for bounded integer in the definition of validity is that the length of the list must be bounded and the elements of the list must be bounded. This shows that this quantification is over a finite set of lists. We will usually use $range(L)$ in constraints to denote a constraints that must be satisfied by all elements of the list. For example, a list L satisfies $range(L) \leq I$ when all elements of the list are smaller than I. Finally, the definition of conversion and subtyping for those new types are defined as expected.

Now we give the typing rules for list constructors in Figure 7. By validity, in the pattern matching rule, the variable a does not appear in Φ_l and Φ' and the variable l does not appear in Φ_a and Φ'. Again, those rules are just a generalization of the ones for integers.

$$\frac{}{\phi;\Phi \vDash \Phi_l\{nil/l\}}$$
$$\frac{}{\phi;\Phi;\Gamma \mid \Theta \vdash_0 \mathtt{nil} : \mathsf{List}(l \mid \Phi_l)}$$
$$\frac{\phi;\Phi;\Gamma \mid \Theta \vdash_R v : \mathsf{Nat}(a \mid \Phi_a) \qquad \phi;\Phi;\Delta \mid \Psi \vdash_{R'} w : \mathsf{List}(l \mid \Phi_l)}{\phi;\Phi;\Gamma,\Delta \mid \Theta + \Psi \vdash_{R+R'} \mathsf{cons}(v,w) : \mathsf{List}(l \mid \Phi_l\{tl(l)/l\}, \Phi_a\{l.(0)/a\}, l \neq nil)}$$
$$\frac{\begin{array}{cc}\phi;(\Phi,\Phi_0\{nil/l\});\Delta \mid \Psi \vdash_R t : \mu & (\phi,a,l);(\Phi,\Phi_0\{cons(a,l)/l\}) \vDash \Phi_a,\Phi_l,\Phi' \\ \phi;\Phi;\Gamma \mid \Theta \vdash_{R'} v : \mathsf{List}(l \mid \Phi_0) & \phi;(\Phi,\Phi');\Delta \mid \Psi \vdash_R w : (\mathsf{Nat}(a \mid \Phi_a) \otimes \mathsf{List}(l \mid \Phi_l)) \multimap \mu\end{array}}{\phi;\Phi;\Gamma,\Delta \mid \Theta + \Psi \vdash_{1+R+R'} \mathtt{match}\ v\ \mathtt{with}\ \{\ \mathtt{nil} \mapsto t \mid \mathtt{cons} \mapsto w\}\ : \mu}$$

Fig. 7. Type System for Dynamic Distribution Types

4.2 Intermediate Terms

We can now detail our example on probabilistic quick sort. We will use $\lambda\langle x,y\rangle.t$ to denote the term $\lambda z.\mathtt{let}\ \langle x,y\rangle = z\ \mathtt{in}\ t$, useful for terms taking a tensor as an input. In the same way, when t is an active term, we write $\mathtt{let}\ \langle x,y\rangle = t\ \mathtt{in}\ u$ for $\mathtt{let}\ z = t\ \mathtt{in}\ \mathtt{let}\ \langle x,y\rangle = z\ \mathtt{in}\ u$.

We remark that for the convolution, if $\mu = \{1 : \sigma \mid b \leq 0\}$, then $\{P : \mu \mid a \leq I\}$ corresponds to $\{P\{c/a\} : \sigma\{c/a\} \mid c \leq I\}$, so it is equal to $\{P : \sigma \mid a \leq I\}$ by renaming. In the same way, the convolution $\{1 : \mu \mid a \leq 0\}$ is μ when a is not free in μ. Thus, we will use a simplified version of the let rules in some derivation tree with those simple convolutions.

Let us recall a useful function symbol that we will use often for primitive recursion: *Bool*. *Bool* is defined by $Bool(0) = 0$ and $Bool(n+1) = 1$, so a recursion context $f : \{Bool(b) : \sigma\{b-1/b\} \mid a \leq 0\}$ expresses that for $b \geq 1$, f is called one time with type $\sigma\{b-1/b\}$ and when $b = 0$, f is never called. This express exactly what happens in primitive recursion. We also define $\neg I = 1 - Bool(I)$.

Subtraction of Integers. Now we can give our general terms working on integers and lists. First we give a term for the subtraction of two integers sub such that sub $\langle n,m\rangle$ is $n-m$. The term is given in Figure 8. For all $\phi;\Phi$ and I,J valid indexes, we have:

$$\phi;\Phi\cdot \vdash_{7\cdot(J+1)} \mathsf{sub} : (\mathsf{Nat}(I) \otimes \mathsf{Nat}(J)) \multimap \mathsf{Nat}(I-J)$$

We give a sketch of the proof. For the sake of simplicity, we always work in the case $\phi;\Phi = \emptyset;\top$. We will use the following notations :
$\sigma = (\mathsf{Nat}(I+b-J) \otimes \mathsf{Nat}(b)) \multimap \mathsf{Nat}(I-J)$
$\Theta = f : \{Bool(b) : \sigma\{b-1/b\} \mid a \leq 0\}$
$\Gamma = n : \mathsf{Nat}(I+b-J), m : \mathsf{Nat}(b)$
$\Delta = \Gamma, p : \mathsf{Nat}(b-1), p' = \mathsf{Nat}(I+b-J-1)$

$$\dfrac{\dfrac{b;(b=0);\Gamma \mid \cdot \vdash_0 n : \mathsf{Nat}(I-J)}{} \qquad \dfrac{\dfrac{\dfrac{\dfrac{b;(b \geq 1, I+b-J \geq 1) \vDash Bool(b) \geq 1}{}}{b;(b \geq 1, I+b-J \geq 1);\Delta \mid \Theta \vdash_0 f\ \langle p',p\rangle : \mathsf{Nat}(I-J)}}{b;(b \geq 1);\Gamma, p : \mathsf{Nat}(b-1) \mid \Theta \vdash_2 \mathtt{match}\ n\ with \cdots : \mathsf{Nat}(I-J)}}{b;(b \geq 1);\Gamma \mid \Theta \vdash_3 \lambda p.\mathtt{match}\ n\ with \cdots : \mathsf{Nat}(a \mid a+1=b) \multimap \mathsf{Nat}(I-J)}}{b;\top;\Gamma \mid \Theta \vdash_4 \mathtt{match}\ m\ \mathtt{with}\ \{\ \mathsf{z} \mapsto\ n \mid \mathsf{s} \mapsto\ \lambda p.\mathtt{match}\ n\ \mathtt{with}\ \{\ \mathsf{z} \mapsto\ \mathsf{z} \mid \mathsf{s} \mapsto\ \lambda p'.f\ \langle p',p\rangle\ \}\ \} : \mathsf{Nat}(I-J)}}{b;\top;\cdot \mid \Theta \vdash_6 \lambda\langle n,m\rangle.\mathtt{match}\ m\ \mathtt{with}\ \{\ \mathsf{z} \mapsto\ n \mid \mathsf{s} \mapsto\ \lambda p.\mathtt{match}\ n\ \mathtt{with}\ \{\ \mathsf{z} \mapsto\ \mathsf{z} \mid \mathsf{s} \mapsto\ \lambda p'.f\ \langle p',p\rangle\ \}\ \} : \sigma}$$
$$\vdash_{7\cdot(J+1)} \mathsf{sub} : \sigma\{J/b\}$$

The final weight $7\cdot(J+1)$ comes from the fact that the index $Q = 7\cdot(b+1)$ satisfies the inequation $b;\top \vDash Q \geq 1 + 6 + Bool(b)\cdot Q\{b-1/b\}$.

Length of a List. The term `length` defined in 8 returns the length of a list. And we have :

$$\phi;\Phi;\cdot \vdash_{6\cdot(J+1)} \texttt{length} : \mathsf{List}(l \mid len(l) = J) \multimap \mathsf{Nat}(J)$$

We use the following notations in the proof :
$\sigma = \mathsf{List}(l \mid len(l) = b) \multimap \mathsf{Nat}(b)$
$\Theta = f : \{Bool(b) : \sigma\{b-1/b\} \mid a' \leq 0\}$
$\Gamma = (hd : \mathsf{Nat}(a \mid \top), tl : \mathsf{List}(l \mid len(l) = b-1))$
$\Delta = (n : \mathsf{Nat}(b-1))$

$$\dfrac{\dfrac{\dfrac{\dfrac{b;(b=0) \vDash b=0}{b;(b=0);\cdot \vdash_0 \mathtt{z} : \mathsf{Nat}(b)} \quad \dfrac{\dfrac{\dfrac{b;(b\geq 1) \vDash Bool(b) \geq 1}{b;(b\geq 1);\Gamma \mid \Theta \vdash_0 f\ tl : Nat(b-1)} \quad \dfrac{}{b;(b\geq 1);\Delta \mid \cdot \vdash_0 \mathtt{s}(n) : \mathsf{Nat}(b)}}{b;(b\geq 1);\Gamma \mid \Theta \vdash_1 \mathtt{let}\ n = (f\ tl)\ \mathtt{in}\ \mathtt{s}(n) : \mathsf{Nat}(b)}}{b;(b\geq 1);\cdot \mid \Theta \vdash_3 \lambda\cdots : (\mathsf{Nat}(a \mid \top) \otimes \mathsf{List}(l \mid len(l) = b-1)) \multimap \mathsf{Nat}(b)} \quad (b,a,l);(b = len(cons(a,l))) \vDash len(l) = b-1, b \geq 1}{b;\top;x : \mathsf{List}(l \mid len(l) = b) \mid \Theta \vdash_4 \mathtt{match}\ x\ \mathtt{with}\ \{\ \mathtt{nil} \mapsto \mathtt{z} \mid \mathtt{cons} \mapsto \ldots\ \} : \mathsf{Nat}(b)}}{b;\top;\cdot \mid f : \{Bool(b) : \sigma\{b-1/b\} \mid 1 \leq a' \leq 1\} \vdash_5 \lambda x.\mathtt{match}\ x\ \mathtt{with}\ \{\ \mathtt{nil} \mapsto \mathtt{z} \mid \mathtt{cons} \mapsto \ldots\ \} : \sigma}}{\vdash_{6\cdot(J+1)} \mathtt{length} : \sigma\{J/b\}}$$

Concatenation of Lists. The term @ denotes concatenation of two lists. We have:
$\phi;\Phi \vdash @ : (\mathsf{List}(l \mid len(l) = J) \otimes \mathsf{List}(l \mid len(l) = I)) \multimap \mathsf{List}(l \mid len(l) = I + J)$.

Let us note $\mathbb{L}(I)$ the type $\mathsf{List}(l \mid len(l) = I)$. We also pose:
$\sigma = (\mathbb{L}(b) \otimes \mathbb{L}(I)) \multimap \mathbb{L}(I+b)$
$\Theta = \{Bool(b) : \sigma\{b-1/b\} \mid a \leq 0\}$
$\tau = (\mathsf{Nat}(a \mid \top) \otimes \mathbb{L}(b-1)) \multimap \mathbb{L}(I+b)$
$\Gamma = x : \mathbb{L}(b), x' : \mathbb{L}(I))$
$\Gamma = x' : \mathbb{L}(I), hd : \mathsf{Nat}(a \mid \top), tl : \mathbb{L}(b-1)$

$$\dfrac{\dfrac{\dfrac{\dfrac{b;(b=0);y : \mathbb{L}(I) \mid \cdot \vdash_0 y : \mathbb{L}(I+b)}{b;\top;\Gamma \mid \Theta \vdash_4 \mathtt{match}\ x\ \mathtt{with}\ \{\ \mathtt{nil} \mapsto x' \mid \mathtt{cons} \mapsto \lambda\langle hd,tl\rangle.\mathtt{let}\ y = f\ \langle tl,x'\rangle\ \mathtt{in}\ \mathtt{cons}(hd,y)\} : \mathbb{L}(I+b)} \quad \dfrac{\dfrac{b;(b\geq 1);\Delta, y : \mathbb{L}(I+b-1) \mid \cdot \vdash_0 \mathtt{cons}(hd,y) : \mathbb{L}(I+b)}{b;(b\geq 1);\Delta \mid \Theta \vdash_1 \mathtt{let}\ y = f\ \langle tl,x'\rangle\ \mathtt{in}\ \mathtt{cons}(hd,y) : \mathbb{L}(I+b)}}{b;(b\geq 1);x' : \mathbb{L}(I) \mid \Theta \vdash_3 \lambda\langle hd,tl\rangle.\mathtt{let}\ y = f\ \langle tl,x'\rangle\ \mathtt{in}\ \mathtt{cons}(hd,y) : \tau}}{b;\top;\cdot \mid \Theta \vdash_6 \lambda\langle x,x'\rangle.\mathtt{match}\ x\ \mathtt{with}\ \{\ \mathtt{nil} \mapsto x' \mid \mathtt{cons} \mapsto \lambda\langle hd,tl\rangle.\ldots.\}\ : (\mathbb{L}(b) \otimes \mathbb{L}(I)) \multimap \mathbb{L}(I+b)}}{\vdash_{7(J+1)} \mathtt{fix}\ f.\lambda\langle x,x'\rangle.\mathtt{match}\ x\ \mathtt{with}\ \{\ \mathtt{nil} \mapsto x' \mid \mathtt{cons} \mapsto \lambda\langle hd,tl\rangle.\mathtt{let}\ y = f\ \langle tl,x'\rangle\ \mathtt{in}\ \mathtt{cons}(hd,y)\}\ : \sigma\{J/b\}}$$

Element of a List. We also have a term `nth` such that for a list l and an integer n, `nth` $\langle n,l\rangle$ returns the n^{th} elements of l (starting from 0) if it exists, and 0 otherwise. The following type can be derived:

$$\phi;\Phi;\cdot \vdash_{8(J+1)} \mathtt{nth} : \forall a : a \leq J.\forall l : (range(l) \leq K, len(l) = J+1).(\mathsf{Nat}(a) \otimes \mathsf{Nat}(l)) \multimap \mathsf{Nat}(l.(a))$$

We use the following notations :
$\Phi_a = (a \leq b)$
$\Phi_l = (range(l) \leq K, len(l) = b+1)$
$\tau = (\mathsf{Nat}(a) \otimes \mathsf{List}(l)) \multimap \mathsf{Nat}(l.(a))$
$\sigma = \forall a : \Phi_a, l : \Phi_l.\tau$
$\Theta = f : \{Bool(b) : \sigma\{b-1/b\} \mid a' \leq 0\}$
$\phi;\Phi = (b,a,l);(\Phi_a,\Phi_l)$
$\Gamma = (n : \mathsf{Nat}(a), x : \mathsf{List}(l))$

$\Phi' = (\Phi, l \neq nil)$
$\Delta = \Gamma, hd : \mathsf{Nat}(l.(0)), tl : \mathsf{List}(tl(l))$

$$\dfrac{\dfrac{\dfrac{\dfrac{\phi, (\Phi', a \geq 1) \vDash Bool(b) \geq 1}{\phi, (\Phi', a \geq 1); \cdot \mid \Theta \vdash_0 f : (\forall a : \Phi_a, l : \Phi_l.\tau)\{b-1/b\}} \quad \phi; (\Phi', a \geq 1) \vDash (a - 1 \leq b - 1), (range(tl(l)) \leq K, len(tl(l)) = b)}{\phi, (\Phi', a \geq 1); \cdot \mid \Theta \vdash_0 f : (\mathsf{Nat}(a-1) \otimes \mathsf{List}(tl(l))) \multimap \mathsf{Nat}(tl(l).(a-1))} \quad \dfrac{}{\phi; (\Phi', a = 0); \Delta \mid \cdot \vdash_0 hd : \mathsf{Nat}(l.(a))}}{\phi; (\Phi', a \geq 1); \Delta \mid \Theta \vdash_1 \lambda p.f\ (p \otimes tl) : \mathsf{Nat}(a' \mid a' + 1 = a) \multimap \mathsf{Nat}(l.(a))}}{\phi; \Phi'; \Delta \mid \Theta \vdash_2 \mathtt{match}\ n\ \mathtt{with}\ \{\ \mathsf{z} \mapsto\ hd \mid \mathsf{s} \mapsto\ \lambda p.f\ \langle p, tl\rangle\ \} : \mathsf{Nat}(l.(a))}$$

$$\dfrac{\phi; (\Phi, l \neq nil); \Gamma \mid \Theta \vdash_4 \lambda\langle hd, tl\rangle.\mathtt{match}\ n\ \mathtt{with}\ \{\ \mathsf{z} \mapsto\ hd \mid \mathsf{s} \mapsto\ \lambda p.f\ \langle p, tl\rangle\ \} : (\mathsf{Nat}(l.(0)) \otimes \mathsf{List}(tl(l))) \multimap \mathsf{Nat}(l.(a))}{(\phi, a', l'); (\Phi, l = cons(a', l')) \vDash a' = l.(0), l' = tl(l), l \neq nil}$$

$$\dfrac{\phi; \Phi; \Gamma \mid \Theta \vdash_5 \mathtt{match}\ x\ \mathtt{with}\ \{\ \mathtt{nil} \mapsto \mathsf{z} \mid \mathtt{cons} \mapsto \lambda\langle hd, tl\rangle.\mathtt{match}\ n\ \mathtt{with}\ \{\ \mathsf{z} \mapsto\ hd \mid \mathsf{s} \mapsto\ \lambda p.f\ \langle p, tl\rangle\ \}\}\ : \mathsf{Nat}(l.(a))}{\phi; \Phi; \cdot \mid \Theta \vdash_7 \lambda\langle n, x\rangle.\mathtt{match}\ x\ \mathtt{with}\ \{\ \mathtt{nil} \mapsto \mathsf{z} \mid \mathtt{cons} \mapsto \lambda\langle hd, tl\rangle.\mathtt{match}\ n\ \mathtt{with}\ \{\ \mathsf{z} \mapsto\ hd \mid \mathsf{s} \mapsto\ \lambda p.f\ \langle p, tl\rangle\ \}\}\ : \tau}$$

$$\dfrac{b; \top; \cdot \mid f : \{Bool(b) : \sigma\{b-1/b\} \mid 1 \leq a' \leq 1\} \vdash_7 \lambda\langle n, x\rangle. \ldots : \forall a : \Phi_a, l : \Phi_l.\tau}{\vdash_{8\cdot(J+1)} \mathtt{nth} : \sigma\{J/b\}}$$

4.3 Partitions and Quick Sort

We can now work on terms specific to quicksort.

Deterministic Partition. The term `partition` is defined such that given an integer n and a list l, `partition` $\langle n, l\rangle$ returns a tensor $\langle\langle l_l, l_r\rangle, p\rangle$ such that $p = l.(n)$ and l_l is the list of all elements in l strictly smaller than p, and l_r is the list of all elements in l strictly greater than p. The term is described in Figure 8. In order to give the type of `partition`, we introduce two notations.

Definition 4.3 (Constraints and Indexes for Lists). For a list index L, we add the new constraint $alldif(L)$, such that $\phi; \Phi \vDash alldif(L)$ if for all valuation $\rho \vDash \Phi$, all elements of the list $[\![L]\!]_\rho$ are different. We also define $(L \subset L')$ such that $(L \subset L')$ is verified when the elements of L are a subset of the elements of L' (taking in account multiplicity). Remark that for any L, L', $(L \subset L')$ and $alldif(L')$ implies $alldif(L)$.

For a list index L and a natural index I, we define the index $\#_{<}(L, I)$ as the number of elements in L strictly smaller than I. In the same way, we also define this index for other relation $>, =, \geq, \ldots$.

From now on, we will note $L \in \mathbb{L}(I)[\leq K]$ to express the set of constraints
$alldif(L), range(L) \leq K, len(L) = I$, that is to say lists of length I, with different elements bounded by K.

Now, for $\phi; \Phi$ and J, K valid natural indexes under $\phi; \Phi$, we have the typing:

$$\phi; \Phi; \cdot \vdash \mathtt{partition} : \forall l : (l \in \mathbb{L}(J+1)[\leq K]).\forall a : (a \leq J).(\mathsf{Nat}(a) \otimes \mathsf{List}(l)) \multimap$$

$$(\mathsf{List}(l' \mid l' \in \mathbb{L}(\#_{<}(l, l.(a)))[\leq K]) \otimes (\mathsf{List}(l' \mid l' \in \mathbb{L}(\#_{>}(l, l.(a))))[\leq K])) \otimes \mathsf{Nat}(a' \mid a' \leq K)$$

Intuitively, this type describe that from a list l of length $J+1$, with different elements bounded by K, and an integer a smaller than J, `partition` returns two lists of different elements bounded by K with a length corresponding to the number of elements strictly smaller (or greater) than $l.(a)$. And the last output of `partition` is an element of the list, ie an integer smaller than K.

We remark that the fact that all elements are different is not primordial for this typing. However, it will be useful later for complexity analysis.

We give a sketch of the typing. We decompose the derivation tree for the sake of clarity. We will use the following notations :

```
sub ≡ fix f. λ⟨n,m⟩.
   match m with
   | 0 ↦ n
   | s ↦ λp. match n with
             | 0 ↦ z
             | s ↦ λp'. f ⟨p',p⟩
length ≡ fix f. λx.
   match x with
   | nil ↦ z
   | cons ↦ λ⟨hd, tl⟩.
      let n = f tl
      in s(n)
@ ≡ fix f. λ⟨x,x'⟩.
   match x with
   | nil ↦ x'
   | cons ↦ λ⟨hd, tl⟩.
      let y = f ⟨tl,x'⟩
      in cons(hd,y)
nth ≡ fix f.λ ⟨n,x⟩.
   match x with
   | nil ↦ z
   | cons ↦ λ⟨hd, tl⟩.
      match n with
      | 0 ↦ hd
      | s ↦ λp. f ⟨p, tl⟩
partition ≡ λ⟨n,x⟩. let p = nth ⟨n,x⟩ in ⟨(walk x),p⟩
 where
   walk ≡ fix f. λy.
      match y with
      | nil ↦ nil ⊗ nil
      | cons ↦ λ⟨hd, tl⟩.
         let ⟨l,r⟩ = f tl
         in let b = sub ⟨p,hd⟩
         in match b with
            | 0 ↦ let b' = sub ⟨hd,p⟩
                  in match b' with
                  | 0 ↦ ⟨l,r⟩
                  | s ↦ λ_. ⟨l,cons(hd,r)⟩
            | s ↦ λ_.⟨cons(hd,l),r⟩
ppart ≡ λx.
    let n = length x
    in match n with
    | 0 ↦ ⟨nil,nil,z⟩
    | s ↦ λm. let i = Unif m in partition ⟨i,x⟩
pquicksort ≡ fix f. λx.
    match x with
    | nil ↦ nil
    | cons ↦ λ_. let ⟨l,r,p⟩ = ppart x
              in let l' = f l
              in let r' = f r
              in l' @ cons(p,r')
```

Fig. 8. Quicksort

$\Phi_l = l \in \mathbb{L}(J+1)[\leq K]$
$\Phi_a = a \leq J$
$\tau_l = \mathsf{List}(l' \mid l' \in \mathbb{L}(\# < (l, l.(a)))[\leq K])$
$\tau_r = \mathsf{List}(l' \mid l' \in \mathbb{L}(\# > (l, l.(a)))[\leq K])$

$\tau_o = (\tau_l \otimes \tau_r) \otimes \mathsf{Nat}(a' \mid a' \leq K)$
$\tau = (\mathsf{Nat}(a) \otimes \mathsf{List}(l)) \multimap \tau_o$
$\phi; \Phi = (l, a); (\Phi_l, \Phi_a)$
$\Gamma' = n : \mathsf{Nat}(a), x : \mathsf{List}(l)$
$\Gamma = \Gamma', p : \mathsf{Nat}(l.(a))$

$$\dfrac{\dfrac{\dfrac{\dfrac{\text{Previous typing with instantiation}}{\phi; \Phi; \cdot \vdash_{8(J+1)} \mathtt{nth} : (\mathsf{Nat}(a) \otimes \mathsf{Nat}(l)) \multimap \mathsf{Nat}(l.(a))} \quad \dfrac{}{\phi; \Phi; \Gamma \mid \cdot \vdash_R \mathtt{walk} : \mathsf{List}(l) \multimap (\tau_l \otimes \tau_r)}}{\phi; \Phi; \Gamma' \mid \cdot \vdash_{8(J+1)} \mathtt{nth}\ \langle n, x\rangle : \mathsf{Nat}(l.(a))} \quad \dfrac{}{\phi; \Phi; \Gamma', p : \mathsf{Nat}(l.(a)) \mid \cdot \vdash_R \langle(\mathtt{walk}\ x), p\rangle : \tau_o}}{\phi; \Phi; n : \mathsf{Nat}(a), x : \mathsf{List}(l) \mid \cdot \vdash_{1+R+8(J+1)} \mathtt{let}\ p = \mathtt{nth}\ \langle n, x\rangle\ \mathtt{in}\ \langle(\mathtt{walk}\ x), p\rangle : \tau_o}}{(a, l); (\Phi_a, \Phi_l); \cdot \vdash_{3+R+8(J+1)} \lambda\langle n, x\rangle.\mathtt{let}\ p = \mathtt{nth}\ \langle n, x\rangle\ \mathtt{in}\ \langle(\mathtt{walk}\ x), p\rangle : (\mathsf{Nat}(a) \otimes \mathsf{List}(l)) \multimap \tau_o}}{\vdash_{1+R+8(J+1)} \mathtt{partition} : \forall l : \Phi_l, a : \Phi_a.\tau}$$

With R the weight of walk, that is to say $(14(K + 1) + 10)(J + 2)$ as shown below. Now we show: $\phi; \Phi; \Gamma \mid \cdot \vdash \mathtt{walk} : \mathsf{List}(l) \multimap (\tau_l \otimes \tau_r)$.

For this typing , we introduce other notations:
$\Phi_{l_0} = l_0 \subset l, range(l_0) \leq K, len(l_0) = b$
$\sigma_l = \mathsf{List}(l' \mid l' \subset l_0, len(l') = \# < (l_0, l.(a)))$
$\sigma_r = \mathsf{List}(l' \mid l' \subset l_0, len(l') = \# > (l_0, l.(a)))$
$\sigma = \forall l_0 : \Phi_{l_0}.\mathsf{List}(l_0) \multimap \sigma_o$
$\Theta = f : \{Bool(b) : \sigma\{b - 1/b\} \mid a' \leq 0\}$
$(\phi, b, l_0); (\Phi; \Phi_{l_0}) = \phi'; \Phi'$
$\mathtt{choose} := \mathtt{let}\ b = \mathtt{sub}\ \langle p, hd\rangle\ \mathtt{in}\ \mathtt{match}\ b\ \mathtt{with}$
$|0 \mapsto \mathtt{let}\ b' = \mathtt{sub}\ \langle hd, p\rangle\ \mathtt{in}\ \mathtt{match}\ b'\ \mathtt{with}\ \{\ \mathtt{z} \mapsto\ \langle l, r\rangle \mid \mathtt{s} \mapsto\ \lambda\_.\langle l, \mathsf{cons}(hd, r)\rangle\ \}$
$|\mathtt{s} \mapsto \lambda\_.\langle \mathsf{cons}(hd, l), r\rangle$

First, remark that $\phi; \Phi \vdash (\mathsf{List}(l_0) \multimap (\sigma_l \otimes \sigma_r))\{l/l_0\} \sqsubseteq \mathsf{List}(l) \multimap (\tau_l \otimes \tau_r)$, indeed as explained previously $alldif(l)$ and $(l' \subset l)$ implies $alldif(l')$. Moreover, $(l' \subset l)$ and $range(l) \leq K$ implies $range(l') \leq K$. So it is sufficient to give a proof of $\phi; \Phi; \Gamma \mid \cdot \vdash \mathtt{walk} : (\mathsf{List}(l_0) \multimap (\sigma_l \otimes \sigma_r))\{l/l_0\}$. Also, remark that $\phi; \Phi \vDash \Phi_{l_0}\{J + 1/b\}\{l/l_0\}$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\pi(f\ tl) \qquad \pi(\mathsf{choose})}{\phi'; (\Phi', l_0 \neq nil); \Gamma, hd : \mathsf{Nat}(l_0.(0)), tl : \mathsf{List}(tl(l_0)) \mid \Theta \vdash_{14(K+1)+6} \mathtt{let}\ \langle l, r\rangle = f\ tl\ \mathtt{in}\ \mathsf{choose} : \sigma_l \otimes \sigma_r}}{\phi'; (\Phi', l_0 \neq nil); \Gamma \mid \Theta \vdash_{14(K+1)+8} \lambda\langle hd, tl\rangle.\mathtt{let}\ \langle l, r\rangle = f\ tl\ \mathtt{in}\ \mathsf{choose} : (\mathsf{Nat}(l_0.(0)) \otimes \mathsf{List}(tl(l_0))) \multimap \sigma_l \otimes \sigma_r}}{\phi'; \Phi'; \Gamma, y : \mathsf{List}(l_0) \mid \Theta \vdash_{14(K+1)+9} \mathtt{match}\ y\ \mathtt{with}\ \{\ \mathtt{nil} \mapsto \langle \mathtt{nil}, \mathtt{nil}\rangle \mid \mathsf{cons} \mapsto \lambda\langle hd, tl\rangle. \ldots .\}\ : \sigma_l \otimes \sigma_r}}{\phi'; \Phi'; \Gamma \mid \Theta \vdash_{14(K+1)+10} \lambda y.\mathtt{match}\ y\ \mathtt{with}\ \{\ \mathtt{nil} \mapsto \langle \mathtt{nil}, \mathtt{nil}\rangle \mid \mathsf{cons} \mapsto \lambda\langle hd, tl\rangle. \ldots .\}\ : \mathsf{List}(l_0) \multimap (\sigma_l \otimes \sigma_r)}}{(\phi, b); \Phi; \Gamma \mid \Theta \vdash_{14(K+1)+10} \lambda y.\mathtt{match}\ y\ \mathtt{with}\ \{\ \mathtt{nil} \mapsto \langle \mathtt{nil}, \mathtt{nil}\rangle \mid \mathsf{cons} \mapsto \lambda\langle hd, tl\rangle. \ldots .\}\ : \sigma}}{\phi; \Phi; \Gamma \mid \cdot \vdash_{(14(K+1)+10)(J+2)} \mathtt{walk} : \forall l_0 : \Phi_{l_0}\{J + 1/b\}.(\mathsf{List}(l_0) \multimap (\sigma_l \otimes \sigma_r)) = \sigma\{J + 1/b\}}}{\phi; \Phi; \Gamma \mid \cdot \vdash_{(14(K+1)+10)(J+2)} \mathtt{walk} : (\mathsf{List}(l_0) \multimap (\sigma_l \otimes \sigma_r))\{l/l_0\}}$$

with $\pi(f\ tl)$ described below :

$$\dfrac{\dfrac{\dfrac{\phi'; (\Phi', l_0 \neq nil) \vDash \Phi_{l_0}\{b - 1/b\}\{tl(l_0)/l_0\} \qquad \dfrac{\phi'; (\Phi', l_0 \neq nil) \vDash Bool(b) \geq 1}{\phi'; (\Phi', l_0 \neq nil); \cdot \mid \Theta \vdash_0 f : \sigma\{b - 1/b\}}}{\phi'; (\Phi', l_0 \neq nil); \cdot \mid \Theta \vdash_0 f : \mathsf{List}(tl(l_0)) \multimap (\sigma_l \otimes \sigma_r)\{tl(l_0)/l_0\} = (\mathsf{List}(l_0) \multimap (\sigma_l \otimes \sigma_r))\{tl(l_0)/l_0\}}}{\phi'; (\Phi', l_0 \neq nil); tl : \mathsf{List}(tl(l_0)) \mid \Theta \vdash_0 f\ tl : (\sigma_l \otimes \sigma_r)\{tl(l_0)/l_0\}}$$

Let us denote $\Delta = \Gamma, hd : \mathsf{Nat}(l_0.(0)), tl : \mathsf{List}(tl(l_0)), l : \sigma_l\{tl(l_0)/l_0\}, r : \sigma_r\{tl(l_0)/l_0\}$. Then, $\pi(\mathsf{choose})$ is a proof of $\phi'; (\Phi', l_0 \neq nil); \Delta \mid \cdot \vdash \mathsf{choose} : \sigma_l \otimes \sigma_r$.

$$\dfrac{\dfrac{\dfrac{\phi'; (\Phi', l_0 \neq nil, l_0.(0) \geq l.(a)); \Delta \mid \cdot \vdash_{7(K+1)+3} t_0 : \sigma_l \otimes \sigma_r \qquad \phi'; (\Phi', l_0 \neq nil, l_0.(0) < l.(a)); \Delta \mid \cdot \vdash_0 t_1 : \sigma_l \otimes \sigma_r}{\phi'; (\Phi', l_0 \neq nil); \Delta, b : \mathsf{Nat}(l.(a) - l_0.(0)) \mid \cdot \vdash_{7(K+1)+4} \mathtt{match}\ b\ \mathtt{with}\ \{\ \mathsf{z} \mapsto\ t_0 \mid \mathsf{s} \mapsto\ \lambda\_.t_1\ \} : \sigma_l \otimes \sigma_r}}{\phi'; (\Phi', l_0 \neq nil); \Delta \mid \cdot \vdash_{7(l_0.(0)+1)+7(K+1)+4} \mathsf{choose} : \sigma_l \otimes \sigma_r}}{\phi'; (\Phi', l_0 \neq nil); \Delta \mid \cdot \vdash_{14(K+1)+4} \mathsf{choose} : \sigma_l \otimes \sigma_r}$$

with $t_0 := \mathtt{let}\ b' = \mathsf{sub}\ \langle hd, p\rangle\ \mathtt{in}\ \mathtt{match}\ b'\ \mathtt{with}\ \{\ \mathsf{z} \mapsto\ \langle l, r\rangle \mid \mathsf{s} \mapsto\ \lambda\_.\langle l, \mathsf{cons}(hd, r)\rangle\ \}$ and $t_1 := \langle \mathsf{cons}(hd, l), r\rangle$. We first show the typing for t_1, and we will use it for its counterparts $\langle l, \mathsf{cons}(hd, r)\rangle$ and $\langle l, r\rangle$. We pose $\Phi_1 = (\Phi', l_0 \neq nil, l_0.(0) < l.(a))$, and we pose $\sigma'_l = \mathsf{List}(l' \mid tl(l') \subset tl(l_0), len(tl(l')) = \# < (tl(l_0), l.(a)), l'.(0) = l_0.(0), l' \neq nil)$. Note that σ'_l is the type appearing in the bottom of the cons typing rule when the head has type $\mathsf{Nat}(l_0.(0))$ and the tail has type $\sigma_l\{tl(l_0)/l_0\}$. Also, remark that $tl(l') \subset tl(l_0)$ and $l'.(0) = l_0.(0)$ implies $l' \subset l_0$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\phi'; \Phi_1 \vDash 1 + \# < (tl(l_0), l.(a)) = \# < (l_0, l.(a))}{\phi'; \Phi_1 \vdash \sigma'_l \sqsubseteq \sigma_l} \qquad \dfrac{\phi'; \Phi_1 \vDash (\# > (l_0, l.(a))) = (\# > (tl(l_0), l.(a)))}{\phi'; \Phi_1 \vdash \sigma_r\{tl(l_0)/l_0\} \sqsubseteq \sigma_r}}{\phi'; \Phi_1; \Delta \mid \cdot \vdash_0 \mathsf{cons}(hd, l) : \sigma_l} \qquad \dfrac{\phi'; \Phi_1; \Delta \mid \cdot \vdash_0 r : \sigma_r}{\phi'; \Phi_1; \Delta \mid \cdot \vdash_0 r : \sigma_r}}{\phi'; (\Phi', l_0 \neq nil, l.(a) > l_0.(0)); \Delta \mid \cdot \vdash_0 t_1 : \sigma_l \otimes \sigma_r}$$

Now we present the typing for t_0, we pose $\Phi_0 = (\Phi', l_0 \neq nil, l_0.(0) \geq l.(a))$.

$$\dfrac{\dfrac{\dfrac{\phi'; (\Phi_0, l_0.(0) \leq l.(a)); \Delta, \mid \cdot \vdash_0 \langle l, r\rangle : \sigma_l \otimes \sigma_r \qquad \phi'; (\Phi_0, l_0.(0) > l.(a)); \Delta \mid \cdot \vdash_0 \langle l, \mathsf{cons}(hd, r)\rangle : \sigma_l \otimes \sigma_r}{\phi'; \Phi_0; \Delta, b' : \mathsf{Nat}(l_0.(0) - l.(a)) \mid \cdot \vdash_2 \mathtt{match}\ b'\ \mathtt{with}\ \{\ \mathsf{z} \mapsto\ \langle l, r\rangle \mid \mathsf{s} \mapsto\ \lambda\_.\langle l, \mathsf{cons}(hd, r)\rangle\ \} : \sigma_l \otimes \sigma_r}}{\phi'; \Phi_0; \Delta \mid \cdot \vdash_{7(l.(a)+1)+3} \mathtt{let}\ b' = \mathsf{sub}\ \langle hd, p\rangle\ \mathtt{in}\ \mathtt{match}\ b'\ \mathtt{with}\ \{\ \mathsf{z} \mapsto\ \langle l, r\rangle \mid \mathsf{s} \mapsto\ \lambda\_.\langle l, \mathsf{cons}(hd, r)\rangle\ \} : \sigma_l \otimes \sigma_r}}{\phi'; \Phi_0; \Delta \mid \cdot \vdash_{7(K+1)+3} \mathtt{let}\ b' = \mathsf{sub}\ \langle hd, p\rangle\ \mathtt{in}\ \mathtt{match}\ b'\ \mathtt{with}\ \{\ \mathsf{z} \mapsto\ \langle l, r\rangle \mid \mathsf{s} \mapsto\ \lambda\_.\langle l, \mathsf{cons}(hd, r)\rangle\ \} : \sigma_l \otimes \sigma_r}$$

And then those two typing for $\langle l, r\rangle$ and $\langle l, \mathsf{cons}(hd, r)\rangle$ are similar to the previous one for t_1.

So, finally, putting it all together, we have indeed the typing
$\vdash \mathtt{partition} : \forall l : (l \in \mathbb{L}(J+1)[\leq K]).\forall a : (a \leq J).(\mathsf{Nat}(a) \otimes \mathsf{List}(l)) \multimap$
$(\mathsf{List}(l' \mid l' \in \mathbb{L}(\# < (l, l.(a)))[\leq K]) \otimes (\mathsf{List}(l' \mid l' \in \mathbb{L}(\# > (l, l.(a))))[\leq K])) \otimes \mathsf{Nat}(a' \mid a' \leq K)$.

The final weight for this proof is $1 + (14(K+1) + 10)(J+2) + 8(J+1)$, that we note $R(K, J+1)$. In this typing, a comparison correspond to the operator sub, that can be given a weight $7(K+1)$ when done between two elements of a list of range smaller than K. So, if we consider the number of comparison, $R(K, J+1)$ shows that the number of comparison in partition is linear in the length of the list.

Partition with Probabilistic Pivot. We can now work on the probabilistic part of our program. The term ppart defined in figure 8 stands for probabilistic partition. The idea for this probabilistic partition is to chose the pivot uniformly at random in the list. Once the pivot has been chosen, we can use the deterministic partition defined above to separate our list. The first direct typing we can give to this term is:

$$\phi; \Phi; \cdot \vdash \mathtt{ppart} : \forall l : (l \in \mathbb{L}(J+1)[\leq K]).\mathsf{List}(l) \multimap$$

$$\{\frac{1}{J+1} : \mathsf{List}(l' \mid l' \in \mathbb{L}(\# < (l, l.(a)))[\leq K]) \otimes \mathsf{List}(l' \mid l' \in \mathbb{L}(\# > (l, l.(a)))[\leq K]) \otimes \mathsf{Nat}(a' \mid a' \leq K) \mid a \leq J\}$$

In this type, the output depends on the choice of the pivot. So we have a distribution over all possible choice of position for the pivot (this corresponds to $0 \leq a \leq J$), and then the output corresponds to the output of `partition` on this choice of pivot. We introduce some notations for the typing derivation :
$\Phi_l = l \in \mathbb{L}(J+1)[\leq K]$
$\sigma_l = \mathsf{List}(l' \mid l' \in \mathbb{L}(\#_< (l, l.(a)))[\leq K])$
$\sigma_r = \mathsf{List}(l' \mid l' \in \mathbb{L}(\#_> (l, l.(a)))[\leq K])$
$\sigma = \sigma_l \otimes \sigma_r \otimes \mathsf{Nat}(a' \mid a' \leq K)$
$\mu = \{\frac{1}{J+1} : \sigma \mid a \leq J\}$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(l,a);(\Phi_l, a \leq J); x : \mathsf{List}(l), i : \mathsf{Nat}(a) \mid \cdot \vdash_{R(K,J)} \mathtt{partition}\ \langle i, x\rangle : \sigma_l \otimes \sigma_r \otimes \mathsf{Nat}(a' \mid a' \leq K)}}{l; \Phi_l; m : \mathsf{Nat}(J) \mid \cdot \vdash_1 \mathtt{Unif}\ m : \{\frac{1}{J+1} : \mathsf{Nat}(a) \mid a \leq J\}}}{l; \Phi_l; x : \mathsf{List}(l), m : \mathsf{Nat}(J) \mid \cdot \vdash_{2+R(K,J)} \mathtt{let}\ i = (\mathtt{Unif}\ m)\ \mathtt{in}\ \mathtt{partition}\ \langle i, x\rangle : \{\frac{1}{J+1} : \sigma \mid a \leq J\}}}{l; \Phi_l; x : \mathsf{List}(l) \mid \cdot \vdash_{3+R(K,J)} \lambda m.\mathtt{let}\ i = (\mathtt{Unif}\ m)\ \mathtt{in}\ \mathtt{partition}\ \langle i, x\rangle : \mathsf{Nat}(J) \multimap \mu}}{l; \Phi_l; x : \mathsf{List}(l), n : \mathsf{Nat}(J+1) \mid \cdot \vdash_{4+R(K,J)} \mathtt{match}\ n\ \mathtt{with}\ \{\ \mathsf{z} \mapsto\ \langle\langle \mathsf{nil}, \mathsf{nil}\rangle, \mathsf{z}\rangle \mid \mathsf{s} \mapsto\ \lambda m. \ldots\ \} : \mu}}{l; \Phi_l; x : \mathsf{List}(l) \mid \cdot \vdash_{5+R(K,J)+6(J+2)} \mathtt{let}\ n = \mathtt{length}\ \mathsf{x}\ \mathtt{in}\ \mathtt{match}\ n\ \mathtt{with}\ \{\ \mathsf{z} \mapsto\ \ldots \mid \mathsf{s} \mapsto\ \ldots\ \} : \mu}}{l; \Phi_l; \cdot \vdash_{6+R(K,J)+6(J+2)} \lambda x.\mathtt{let}\ n = \mathtt{length}\ x\ \mathtt{in}\ \ldots : \mathsf{List}(l) \multimap \mu}}{\vdash_{6+R(K,J)+6(J+2)} \mathtt{ppart} : \forall l : \Phi_l.\mathsf{List}(l) \multimap \mu}$$

And those two leaves come from the typing of `Unif` and the previous typing of `partition`.

We will note $R'(K, J+1) = 6 + R(K, J+1) + 6(J+2)$. Again, this index has the form $n_{0,0} + n_{1,0}K + n_{0,1}(J+1) + n_{1,1}K \cdot (J+1)$, and so the number of comparison in `ppart` is linear in the length of the list.

However, we want to give a more satisfactory typing to `ppart` using subtyping. Let us pose:
$\tau_l = \mathsf{List}(l' \mid l' \in \mathbb{L}(c)[\leq K])$
$\tau_r = \mathsf{List}(l' \mid l' \in \mathbb{L}(J-c)[\leq K])$
$\tau = \tau_l \otimes \tau_r \otimes \mathsf{Nat}(a' \mid a' \leq K)$
$\nu = \{\frac{1}{J+1} : \tau \mid c \leq J\}$. Note that in ν, the list index l does not appear.

And we can derive:

$$\dfrac{\dfrac{}{\mathsf{List}(l \mid \Phi_l) \triangleright l : \Phi_l.\mathsf{List}(l)} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{l; \Phi_l \vdash \mathsf{List}(l) \sqsubseteq \mathsf{List}(l)} \qquad l; \Phi_l \vdash \mu \sqsubseteq \nu}{l; \Phi_l \vdash \mathsf{List}(l) \multimap \mu \sqsubseteq \mathsf{List}(l) \multimap \nu}}{l; \Phi_l \vDash \Phi_l \qquad l; \Phi_l \vdash \forall l : \Phi_l.\mathsf{List}(l) \multimap \mu \sqsubseteq \mathsf{List}(l) \multimap \nu}}{\vdash \forall l : \Phi_l.\mathsf{List}(l) \multimap \mu \sqsubseteq \forall l : \Phi_l.\mathsf{List}(l) \multimap \nu}}{\vdash \forall l : \Phi_l.\mathsf{List}(l) \multimap \mu \sqsubseteq \mathsf{List}(l \mid \Phi_l) \multimap \nu}$$

We need to prove $l; \Phi_l \vdash \mu \sqsubseteq \nu$.

We define

$$S = \frac{if(\#_< (l, l.(a)) = c)\ then\ 1\ else\ 0}{J+1}$$

We can show that $l; \Phi_l \vdash S \lhd_\sqsubseteq \langle \mu \& \nu \rangle$

- S is indeed valid under $(l, a, c); (l \in \mathbb{L}(J+1)[\leq K], a \leq J, c \leq J)$.
- Let us show that $(l, c); (l \in \mathbb{L}(J+1)[\leq K], c \leq J) \vDash \sum_{a \leq J} S = \frac{1}{J+1}$. Let ρ be a valuation such that $\rho(l) \in \mathbb{L}([\![J]\!]_\rho + 1)[\leq [\![K]\!]_\rho]$ and $\rho(c) \leq [\![J]\!]_\rho$. Then, $[\![\sum_{a \leq J} S]\!]_\rho = \sum_{n \leq [\![J]\!]_\rho} [\![S]\!]_{\rho[a \mapsto n]}$, and

$[\![S]\!]_{\rho[a\mapsto n]} = \frac{if(\#<(\rho(l),(\rho(l)).(n))=\rho(c))\ then\ 1\ else\ 0}{[\![J]\!]_\rho+1}$. In the list $\rho(l)$, as all elements are different and $len(\rho(l)) = [\![J]\!]_\rho + 1$, there is a **unique** element in the list such that there is exactly $\rho(c)$ elements strictly smaller than it. Let us call n_0 its position in $\rho(l)$. By definition, $n_0 \leq [\![J]\!]_\rho$ and $\# < (\rho(l), (\rho(l)).(n_0)) = \rho(c)$, and for all other $n \neq n_0$, with $n \leq [\![J]\!]_\rho$, this last equality is not verified by unicity. This concludes the proof for this point.

- Let us show that $(l, a); (l \in \mathbb{L}(J+1)[\leq K], a \leq J) \vDash \sum_{c\leq J} S = \frac{1}{J+1}$. This time it is straightforward, the unique c smaller than J verifying $(\# < (l, l.(a))) = c)$ is $(\# < (l, l.(a)))$
- Let us show that $(l, a, c); (l \in \mathbb{L}(J+1)[\leq K], a \leq J, c \leq J, S \neq 0) \vdash \sigma \sqsubseteq \tau$. By definition, $S \neq 0$ is equivalent to $(\# < (l, l.(a)) = c)$. Moreover, as $l \in \mathbb{L}(J+1)[\leq K]$, this equality implies $(\# > (l, l.(a)) = J - c)$. Thus under those conditions, we have $\sigma = \tau$, and this concludes the proof.

So finally, by subtyping, we obtain

$$\phi; \Phi; \cdot \vdash_{R'(K,J+1)} \texttt{ppart} : \mathsf{List}(l \mid \Phi_l) \multimap \nu$$

That is to say

$$\phi; \Phi; \cdot \vdash_{R'(K,J+1)} \texttt{ppart} : \mathsf{List}(l \mid l \in \mathbb{L}(J+1)[\leq K]) \multimap$$

$$\{\frac{1}{J+1} : \mathsf{List}(l' \mid l' \in \mathbb{L}(c)[\leq K]) \otimes \mathsf{List}(l' \mid l' \in \mathbb{L}(J-c)[\leq K]) \otimes \mathsf{Nat}(a' \mid a' \leq K) \mid c \leq J\}$$

And this shows the main goal of probabilistic quick sort : this random choice of pivot guarantees that you obtain uniformly all the different length of partitions.

Probabilistic Quicksort. We can now give the final typing for probabilistic quicksort. The term is described in Figure 8. And we have the typing:

$$\phi; \Phi; \cdot \vdash \texttt{pquicksort} : \mathsf{List}(l \mid l \in \mathbb{L}(J)[\leq K]) \multimap \mathsf{List}(l \mid len(l) = J, range(l) \leq K)$$

We give with some informality a sketch of this proof when $\phi; \Phi = \emptyset; \top$. We pose
$\sigma_0 = \mathsf{List}(l \mid l \in \mathbb{L}(b)[\leq K])$
$\sigma_1 = \mathsf{List}(l \mid len(l) = b, range(l) \leq K)$
$\sigma = \sigma_0 \multimap \sigma_1$
$\Theta = f : \{(if\ (b = 0)\ then\ 0\ else\ \frac{2}{b}) : \sigma\{a/b\} \mid a \leq b-1\}$
$\Gamma = l : \mathsf{List}(l' \mid l' \in \mathbb{L}(c)[\leq K]), r : \mathsf{List}(l' \mid l' \in \mathbb{L}(b-1-c)[\leq K]), p : \mathsf{Nat}(a' \mid a' \leq K)$
$\phi; \Phi = (b, c); (b \geq 1, c \leq b-1)$
$\Psi = f : \{1 : \sigma\{c/b\}\} + \{1 : \sigma\{b-1-c/b\}\}$
$\Psi_1 = f : \{1 : \sigma\{c/b\}\}$
$\Psi_2 = f : \{1 : \sigma\{c/b-1-c\}\}$
$\Delta = \Gamma, l' : \sigma_1\{c/b\}, r' : \sigma_1\{b-1-c/b\}$
We will prove informally later that $b; (b \geq 1) \vdash \Theta \sqsubseteq \{\frac{1}{b} : \Psi \mid c \leq b-1\}$.

$$\dfrac{\dfrac{\phi;\Phi \vDash \neg(0) \geq 1}{\phi;\Phi;\Gamma \mid \Psi_1 \vdash_0 f\ l : \sigma_1\{c/b\}} \quad \dfrac{\dfrac{\phi;\Phi \vDash c+1+b-1-c=b}{\phi;\Phi;\Delta \mid \cdot \vdash_{7c} l'@\mathsf{cons}(p,r') : \sigma_0}}{\phi;\Phi;\Gamma, l' : \sigma_1\{c/b\} \mid \Psi_2 \vdash_{7b+1} \mathtt{let}\ r' = \cdots : \sigma_1}}{(b,c);(b \geq 1, c \leq b-1);\Gamma \mid \Psi \vdash_{7b+2} \mathtt{let}\ l' = f\ l\ \mathtt{in}\ \mathtt{let}\ r' = f\ r\ \mathtt{in}\ l'@\mathsf{cons}(p,r') : \sigma_1}$$

$$\dfrac{b;(b \geq 1);x : \sigma_0 \mid f : \{\frac{1}{b} : \mu \mid c \leq b-1\} \vdash_{7b+6+R'(K,b)} \mathtt{let}\ \langle l,r,p\rangle = \mathtt{ppart}\ x\ \mathtt{in}\ \mathtt{let}\ l' = f\ l\ \mathtt{in}\ \ldots : \sigma_1}{b;(b \geq 1);x : \sigma_0 \mid \Theta \vdash_{7b+6+R'(K,b)} \mathtt{let}\ \langle l,r,p\rangle = \mathtt{ppart}\ x\ \mathtt{in}\ \mathtt{let}\ l' = f\ l\ \mathtt{in}\ \mathtt{let}\ r' = f\ r\ \mathtt{in}\ l'@\mathsf{cons}(p,r') : \sigma_1}$$

$$\dfrac{b;\top;x : \sigma_0 \mid \Theta \vdash_{7b+8+R'(K,b)} \mathtt{match}\ x\ \mathtt{with}\ \{\ \mathtt{nil} \mapsto \mathtt{nil} \mid \mathtt{cons} \mapsto \lambda\_.\mathtt{let}\ \langle l,r,p\rangle = \mathtt{ppart}\ x\ \mathtt{in}\ \ldots\} : \sigma_1}{b;\top;\cdot \mid \Theta \vdash_{7b+9+R'(K,b)} \lambda x.\mathtt{match}\ x\ \mathtt{with}\ \{\ \mathtt{nil} \mapsto \mathtt{nil} \mid \mathtt{cons} \mapsto \ldots\} : \sigma_0 \multimap \sigma_1}$$

$$\vdash_{Q\{J/b\}} \mathtt{pquicksort} : \sigma\{J/b\}$$

The inequations that Q must verify are :

$$\vDash Q\{0/b\} \geq 10 + R'(K,0)$$

$$b;(b \geq 1) \vDash Q \geq 7b + 10 + R'(K,b) + \sum_{a \leq b-1} \frac{2 \cdot Q\{a/b\}}{b}$$

So, intuitively

$$\forall n \in \mathbb{N}, Q(n+1) \geq 7n + 17 + R'(K,n+1) + \sum_{m \leq n} \frac{2 \cdot Q(m)}{n+1}$$

As $R'(K,n+1)$ is linear in n in the number of comparison, we obtain an equation very similar to the usual one for randomized quicksort, thus we can give Q a value with a form $(C \cdot b \cdot log(b)) \cdot K$, expressing the usual complexity of quicksort in the number of comparison.

We show informally that $b;(b \geq 1) \vdash \Theta \sim \{\frac{1}{b} : \Psi \mid c \leq b-1\}$.

$$\Psi = f : \{1 : \sigma\{c/b\}\} + \{1 : \sigma\{b-1-c/b\}\}$$

$$\{\frac{1}{b} : \Psi \mid c \leq b-1\} \sim f : \{\frac{1}{b} : \sigma\{c/b\} \mid c \leq b-1\} + \{\frac{1}{b} : \sigma\{b-1-c/b\} \mid c \leq b-1\}$$

$$\{\frac{1}{b} : \Psi \mid c \leq b-1\} \sim f : \{\frac{1}{b} : \sigma\{c/b\} \mid c \leq b-1\} + \{\frac{1}{b} : \sigma\{c/b\} \mid c \leq b-1\}$$

$$\{\frac{1}{b} : \Psi \mid c \leq b-1\} \sim f : \{\frac{2}{b} : \sigma\{a/b\} \mid a \leq b-1\} = \Theta$$

Thus, our type system can effectively show that the average number of comparison in probabilistic quicksort is in $O(n \log(n)$.

5 EXTENSIONAL COMPLETENESS

5.1 Probabilistic Turing Machines and Their Enconding

Let us express a kinf of completeness for this calculus. In this section, we prove that we can simulate any PAST probabilistic Turing machine. First, let us define what are the kind of Turing machine we will encode.

Definition 5.1 (Probabilistic Turing Machine). A *probabilistic Turing machine* is a tuple $(Q, q_i, \delta_0, \delta_1, F)$ where Q is a set of states, q_i is the initial state, δ_0 and δ_1 are transition functions with type $Q \times \{0,1\} \to Q \times \{0,1\} \times \{left, right, stay\}$ and $F \subseteq Q$ is the set of terminating states. With this definition, we consider that at each step, the Turing machine does a probabilistic choice: with probability $\frac{1}{2}$, it does the δ_0 transition, and with probability $\frac{1}{2}$ it does the δ_1 transition. A configuration of such a Turing machine is a tuple (w, b, w', q) where $b \in \{0,1\}$ is the value of the tape at the current position of the head, and w (resp. w') is the tape before (resp. after) the head. And finally, q

```
init ≡ λw. ⟨ϵ, w, qi⟩

PTM ≡ λwi.
    (fix f. λc. let ⟨w, v, q⟩ = c in
        let r = Unif 1 in
        match r with
            |0 ↦ case(w, v, q, 0)
            |s ↦ λ_. case(w, v, q, 1)
    ) (init wi)
```

Fig. 9. Terms for Turing Machines

represent the current state of the machine. In order for this machine to compute a word, we ask that when the machine reaches a terminating state, then the configuration has the form (ϵ, b, w, q_F) with $q_F \in F$, and the output word is bw. Note that a probabilistic Turing machine computes a distribution of words, since each step is a probabilistic choice. In order to talk about the notion of termination, let us introduce some notations: given a configuration $c = (w, b, w', q)$, we note $next_0(c)$ and $next_1(c)$ the two configurations obtained after a δ_0 or δ_1 transition from c.

We can now define what is a PAST Turing machine. In order to do that, we use lyapunov ranking function.

Definition 5.2 (PAST Turing machine). A probabilistic Turing machine $(Q, q_0, \delta_0, \delta_1, F)$ is PAST when there exists a function f from configurations to rationals such that for any non-final configuration $c = (w, b, w', q)$, we have $f(c) \geq 1 + \frac{f(next_0(c))}{2} + \frac{f(next_1(c))}{2}$, and for all terminating configuration, $f(c) = 0$. Intuitively, this means that $f(c)$ is an upper bound of the expected number of steps from the configuration c. And so, for any initial configuration $c_i = (\epsilon, b, w, q_i)$, we know that in average, the probabilistic Turing machine will stop after $f(c_i)$ steps.

Now we will encode those kind of Turing machine in our calculus. In order to do that, we will represent a configuration (w, b, w', q) by three binary words w_l, w_r, w_q such that w_l is the mirror of w, $w_r = bw'$ and w_q is a word representing the state q, of fixed size S_Q depending of the size of Q. In the same way, we can encode a configuration in an integer, by using for example integers in base 3 and using the digit 2 to separate binary words. Let us introduce some notation for those encoding.

Definition 5.3 (Notations). From now on, configurations will always be represented by the triple (w_l, w_r, w_q). From such a configuration c, we note $Intconf(c)$ the integer representing this configuration. When n is an integer representing a configuration, we denote $\pi_r(n)$, $\pi_l(n)$ and $\pi_q(n)$ the three projections of n into the words w_l, w_r and w_q. Otherwise, when n does not represent a configuration, we send it to a fixed chosen configuration.

So, in our calculus, a configuration (w_l, w_r, w_q) has the type $\mathrm{Word}(w_l) \otimes \mathrm{Word}(w_r) \otimes \mathrm{Word}(w \mid w = w_q \wedge |w| = S_Q)$. Thus, for an natural index I, we denote by $\mathrm{Conf}(I)$ the type $\mathrm{Word}(\pi_l(I)) \otimes \mathrm{Word}(\pi_r(I)) \otimes \mathrm{Word}(w \mid w = \pi_q(I) \wedge |w| = S_Q)$.

5.2 Terms to Simulate Turing Machines

We now describe how to simulate a probabilistic Turing machine $(Q, q_i, \delta_0, \delta_1, F)$ in our calculus. We add a base type to our calculus : binary words with type $\mathrm{Word}(\omega \mid \Phi)$. Thus, we have constructors for successors s_0 and s_1, the empty word ϵ and a match on words. Formally, we should give the typing rule for words, however they can easily be deduced from the one for integers and lists.

To simulate Turing machine, first, in figure 9, we give the term init. Note that in this term, q_i denote the word of fixed size corresponding to the initial state q_i of the Turing machine. We can easily show that init has type $\mathsf{Word}(w) \multimap \mathsf{Word}(\epsilon) \otimes \mathsf{Word}(w) \otimes \mathsf{Word}(w \mid w = q_i \wedge |w| = S_Q)$. Then, to encode the Turing machine, we use the term PTM. Let us describe informally what is the term $\mathsf{case}(w, v, q, 0)$. This term will simulate a δ_0 transition and then apply f to this new configuration. For that, we first recover the first letter of v with a match, and then we decompose entirely q using S_Q match. Then, on a specific conditional branch where we know the first letter b of v (and so $v = \mathsf{s}_b(v')$) and the actual state $q \notin F$, we write a term corresponding to the transition δ_0. For example, if $\delta(q, b) = (q', b', right)$, the term in the conditional branch corresponding to b, q will be $f\ \langle \mathsf{s}_{b'}(w), v', q' \rangle$. Note that we do a recursive call to f in this case in order to continue the computation. However, in the case of a terminating state $q \in F$, we just output the word without calling f, thus the term in a branch for a terminating state is just v since when a Turing machine terminates, the output of this machine is just the current word after the head. This correspond to the base case of our fixpoint. In the same way, the second case simulates δ_1 transitions.

With this definition, from a word w, $PTM\ w$ behaves like the associated Turing machine. Let us show that we can indeed type this term.

5.3 Typing and Complexity Bound

As expressed before, init has type $\mathsf{Word}(w) \multimap \mathsf{Word}(\epsilon) \otimes \mathsf{Word}(w) \otimes \mathsf{Word}(w \mid w = q_i \wedge |w| = S_Q)$. The index variable b in the fixpoint here represents the current configuration, and we obtain :

$$\frac{b; \cdot; \cdot \mid f : \{\frac{active(b)}{2} : \mathsf{Conf}(next_a(b)) \multimap \mathsf{Word}(w \mid \top) \mid a \leq 1\} \vdash_{k_{step}} \lambda c. \cdots : \mathsf{Conf}(b) \multimap \mathsf{Word}(w \mid \top)}{\cdot \vdash_{R\{Intconf(\epsilon, w, q_i)/b\}} \mathtt{fix}\ f.\lambda c. \ldots : \mathsf{Conf}(Intconf(\epsilon, w, q_i)) \multimap \mathsf{Word}(w \mid \top)}$$

The index $active(I)$ is a function equals to O if the configuration encoded by I is final and it is equal to 1 otherwise. Intuitively, this types means that for final configurations, we do not call f, and for active configurations, with probability $\frac{1}{2}$, f is called on the configuration obtained with a δ_0 transition and with probability $\frac{1}{2}$, f is called on the configuration obtained with a δ_1 transition.

Let us sketch how this typing works in the derivation. In the subterm `let r = Unif1 in ...`, we can decompose our distribution for f in two. Indeed, r has type $\mathsf{Nat}(c)$ (with probability $\frac{1}{2}$) for some $c \leq 1$ and for this c, we take the typing hypothesis $v = f : \{active(b) : \mathsf{Conf}(next_c(b)) \multimap \mathsf{Word}(w \mid \top) \mid a \leq 0\}$, and it's direct that the convolution $\{\frac{1}{2} : v \mid c \leq 1\}$ is equal to the original distribution for f. Now, let us take a particular branch of the case term corresponding to the transition $\delta(q, i) = (q', i', right)$. We have to derive the following judgment: $\phi; \Phi; \Gamma \mid \Theta \vdash f\ \langle s_{i'}(w), v', q' \rangle : \mathsf{Word}(w \mid \top)$ with

- $\phi = (b, c)$, where b encode the current configuration and c represents the integer r
- $\Phi = (c = 0, \pi_q(b) = q, (\pi_r(b))_0 = i)$, indeed, with the match on r, on v and on q, we can derive those hypothesis
- $\Gamma = w : \mathsf{Word}(\pi_l(b)), v' : \mathsf{Word}(w \mid s_i(w) = \pi_r(b))$
- $\Theta = f : \{active(b) : \mathsf{Conf}(next_c(b)) \multimap \mathsf{Word}(w \mid \top)\}$

As we supposed that q is not a final state, we know that $active(b) = 1$. Also, knowing that $\pi_q(b) = q$ and $(\pi_r(b))_0 = i$ we can compute $next_0(b)$ and verify that $\mathsf{Conf}(next_0(b))$ is indeed the type of $\langle s_{i'}(w), v', q' \rangle$. In the same way, we can show that the term that simulates the Turing machine is well typed for all other cases.

Now, let us talk about the complexity bound of this term. Let us call f_T the lyapunov ranking function associated to the PAST probabilistic Turing machine. This function f_T is given as a function index F_T from integer to rational numbers such that $F_T(Intconf(w_l, w_r, w_q) = f_T(w_l, w_r, w_q)$. Let

us call k_{step} and k_{init} the constant bounds derived from the typing of $(\lambda c. \dots)$ and (init w_i). We show that the index $R = (1 + k_{step})(F_T(b) + 1)$ satisfies the inequation for the fixpoint. Indeed, R must satisfy

$$b; \cdot \vDash R \geq 1 + k_{step} + \frac{active(b)}{2} R\{next_0(b)/b\} + \frac{active(b)}{2} R\{next_1(b)/b\}$$

When b is a final state, this is direct as $R = 1 + k_{step} \geq 1 + k_{step}$. Otherwise, we obtain:

$$b; \cdot \vDash R \geq 1 + k_{step} + \frac{1}{2} R\{next_0(b)/b\} + \frac{1}{2} R\{next_1(b)/b\}$$

$$b; \cdot \vDash F_T(b) + 1 \geq 1 + \frac{1}{2}(F_T(next_0(b)) + 1) + \frac{1}{2}(F_T(next_1(b)) + 1)$$

$$b; \cdot \vDash F_T(b) \geq 1 + \frac{1}{2} F_T(next_0(b)) + \frac{1}{2} F_T(next_1(b))$$

which is true by definition of the lyapunov ranking function F_T. Thus, to conclude, for the term PTM we obtain the bound $(1 + k_{step})(F_T(Intconf(\epsilon, w, q_i)) + 1) + 1 + k_{init}$. Note that if $f_T(\epsilon, w, q_i)$ is a $O(E(w))$, where $E(w)$ is the expected complexity of the run of the PTM on w, then our bound is also a $O(E(w))$

6 CONCLUSIONS

REFERENCES

Martin Avanzini, Ugo Dal Lago, and Akihisa Yamada. 2018. On Probabilistic Term Rewriting. In *Proc. of 14th FLOPS (LNCS)*, Vol. 10818. Springer, 132–148.

Gilles Barthe, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. 2017. Coupling proofs are probabilistic product programs. In *ACM SIGPLAN Notices*, Vol. 52. ACM, 161–174.

Ugo Dal Lago and Charles Grellois. 2017. Probabilistic Termination by Monadic Affine Sized Typing. In *Proc. of 26th ESOP*. 393–419.

U. Dal Lago and B. Petit. 2012. Linear Dependent Types in a Call-by-value Scenario. In *Proc. of 12th PPDP*. ACM, 115–126.