MPRI - Examen Langage de Programmation Probabiliste
Mars 2023

Consignes Vous avez 3h pour compléter l'examen. Il y a 2 exercices indépendants. Le sujet est long, pensez à aborder les deux parties. Merci de respecter les consignes suivantes :

– Tous les appareils électroniques doivent être éteints et rangés dans votre sac.

– Documentation autorisée: polycopié de cours.

– Le barème est donné à titre indicatif et peut être modifié.

Exercice 1 : Inférence semi-symbolique - 10 points

Calculer exactement la distribution a posteriori décrite par un modèle est généralement un problème très difficile. Les méthodes d'inférences particulaires permettent d'approximer le résultat en rejouant le modèle de nombreuse fois de manière indépendante. Le calcul exact reste cependant possible pour les modèles les plus simples, ou certains sous-termes au sein d'un modèle complexe. Le but de cet exercice est d'implémenter une méthode d'inférence semi-symbolique qui combine calcul symbolique exact et méthode particulaire approchée pour calculer la distribution a posteriori décrite par un modèle.

A. Préambule

Considérons les deux programmes suivants où `a`, `b`, `mu0`, `sigma0`, `sigma` sont des constantes.

```
let beta_bernoulli v =
  let p = sample (beta ~a ~b) in (* p ~ Beta(a, b) *)
  observe (bernoulli ~p) v;      (* v ~ Bernoulli(p) *)
  p

let gauss_gauss v =
  let mu = sample (gaussian ~mu:0. ~sigma:1.) in (* mu ~ N(0, 1) *)
  observe (gaussian ~mu:mu ~sigma:1.) v;         (* v ~ N(mu, 1) *)
  mu
```

Question 1. [*Cours*] En utilisant la sémantique par noyaux vue en cours (cf. annexe), montrer que :

– $[\![\texttt{infer beta\_bernoulli}\ v]\!] = Beta(a + (1 - v),\ b + v)$

– $[\![\texttt{infer gauss\_gauss}\ v]\!] = \mathcal{N}(m, s)$ avec $m = v/2$ et $s^2 = 1/2$

Plus généralement, si la distribution a posteriori est de la même famille que la distribution a priori après une observation, on dit que la distribution a priori et la distribution observée sont *conjugées*. On a ainsi :

– si $\begin{cases} X \sim Beta(a,\ b) \\ Y \sim Bernoulli(X) \end{cases}$ alors $\begin{cases} (X \mid Y = v) \sim Beta(a + (1 - v),\ b + v) \\ Y \sim Bernoulli\left(\dfrac{a}{a+b}\right) \end{cases}$

– si $\begin{cases} X \sim \mathcal{N}(\mu_0,\ \sigma_0) \\ Y \sim \mathcal{N}(X, \sigma) \end{cases}$ alors $\begin{cases} (X \mid Y = v) \sim \mathcal{N}(\mu_1,\ \sigma_1) \\ Y \sim \mathcal{N}(\mu_0, \sqrt{\sigma_0^2 + \sigma^2}) \end{cases}$ avec $\begin{cases} \mu_1 = \left(\dfrac{\mu_0}{\sigma_0^2} + \dfrac{v}{\sigma^2}\right) \Big/ \left(\dfrac{1}{\sigma_0^2} + \dfrac{1}{\sigma^2}\right) \\ \sigma_1^2 = 1 \Big/ \left(\dfrac{1}{\sigma_0^2} + \dfrac{1}{\sigma^2}\right) \end{cases}$

B. Calcul symbolique

On considère le type de distribution suivant.

```
type dist =
  | Beta of node * node
  | Bernoulli of node
  | Gaussian of node * node
  | V of float

and node = dist ref
```

Les paramètres de chaque distribution sont des références qui peuvent être mises à jour pendant l'inférences. Pour simplifier, les valeurs concrètes sont toujours de type `float`.

Question 2. Donner l'implémentation des fonctions suivantes.

```
val const : float → node
(* build a constant node *)

exception Not_tractable

val value : node → float
(* return the value of a constant node
   raise [Not_tractable] if the node is not constant. *)
```

Il faut maintenant définir les opérateurs probabilistes qui effectuent le calcul symbolique. `sample` associe simplement une variable à une distribution (sans tirer de valeur aléatoire), et `observe` met à jour les paramètres d'une distribution étant donné une observation.

```
val sample : dist → node
(* return a new node with distribution [dist] *)

val observe : dist → float → unit
(* update distribution [dist] given an observation
   raise [Not_tractable] if this is not possible. *)

val infer : (α → node) → α → dist
(* return the posterior distribution of a model
   we assume that a model always returns a value of type [node]. *)
```

Question 3. En utilisant les résultats de la section A, donner l'implémentation de ces opérateurs. On pourra compléter le squelette suivant pour `observe`.

```
let observe (d : dist) (x : float) : unit =
  match d with
  | Bernoulli p → (
      match !p with
      | Beta (a, b) → (* Symbolic computation is possible *)
          a := ... (* TODO *)
          b := ... (* TODO *)
      | _ → raise Not_tractable)
  | Gaussian (m, s) → ... (* TODO *)
  | _ → raise Not_tractable
```

Nous avons maintenant un moteur d'inférence symbolique qui permet déjà de traiter les exemples de la section A. Malheureusement, cette implémentation ne fonctionne que pour très peu de modèles.

Question 4. Donner un exemple (simple) de modèle pour lequel l'inférence échoue.

C. Inférence semi-symbolique

Pour traiter plus de cas, l'idée de l'inférence semi-symbolique est de combiner le calcul symbolique avec une méthode particulaire approchée. Si le calcul symbolique échoue on tire aléatoirement des valeurs concrètes pour certaines variables aléatoires avant de continuer l'exécution. Pour cet exercice, nous utilisons *Importance Sampling*.

Question 5. [*Cours*] Rappeler l'implémentation de `infer` (avec un argument `prob` explicite).

```
type prob : {id: int; scores: float array}
val infer : (prob → α → β) → α → β Distribution.t
```

Note : l'opérateur `infer` est polymorphe. Nous pouvons donc l'utiliser avec des modèles de type `prob` $\rightarrow \alpha \rightarrow$ `node`. On pourra utiliser la librairie `Distribution` de BYO-PPL.

```
type α t
(* distribution type *)

val bernoulli ~p: float → float t
val gaussian ~mu:float → ~sigma:float → float t
val dirac ~v:α → α t
val support ~values:α array → ~logits: float array → α t

val draw : α t → α
(* draw a sample from a distribution *)

val logpdf : α t → α → float
(* [logpdf d x]: returns the value of the log-density of [dist] on [x]. *)
```

L'implémentation de `infer` suppose que les opérateurs probabilistes `sample` et `observe` prennent un argument supplémentaire `prob` pour enregistrer le score des particules. L'opérateur `sample` est similaire à celui de la Question 3, et ne modifie pas `prob`.

```
val sample : prob → dist → node
(* return a new node with distribution [dist] *)
```

L'évaluation de `observe` met à jour le score des particules. Une implémentation simple de `observe` convertit une distribution de type `dist` en une distribution classiques qu'on peut ensuite utiliser pour mettre à jour le score.

```
let observe prob (d : dist) (x : float) : unit =
  let marginal = (* convert [dist] to a classic distribution *)
    match d with
    | Bernoulli p → bernoulli ~p:(value p)
    | Gaussian (m, s) → gaussian ~mu:(value m) ~sigma:(value s)
    | Beta (a, b) → beta ~a:(value a) ~b:(value b)
    | V v → dirac ~v
  in (* update the score of particle prob.id *)
  prob.scores.(prob.id) ← prob.scores.(prob.id) +. logpdf marginal x
```

On utilise l'opérateur `value` pour calculer une valeur concrète d'un `node` pour instancier les distributions classiques. En pratique, 1) on échantillonne récursivement tous les ancêtres d'un `node` pour obtenir une valeur concrète des paramètres, 2) on tire aléatoirement une valeur v dans la distribution classique correspondante, 3) on fixe la valeur du `node` à Vv, 3) on renvoie la valeur v.

Question 6. Compléter le code `value` pour implémenter ce comportement.

```
let rec value node : float =
let v = (* compute a concrete value *)
  match !node with
```

```
  | Beta (a, b) → ... (* TODO *)
  | Bernoulli p → .. (* TODO *)
  | Gaussian (mu, sigma) → .. (* TODO *)
  | V v → v
in
node := V v; (* turn the node to a constant *)
v (* return the value *)
```

Question 7. Que se passe-t-il si on appelle `value` deux fois sur le même node ?

On peut maintenant ajouter le calcul symbolique de la section B, à notre opérateur `observe`. Lorsque le calcul symbolique est possible, on met à jour la distribution symbolique avant de renvoyer la distribution concrète qui permet de calculer le score (voir section A).

Question 8. Compléter le code suivant pour implémenter ce comportement.

```
let observe prob (d : dist) (x : float) : unit =
  let marginal = (* convert [dist] to a classic distribution *)
    match d with
    | Bernoulli p → (
        match !p with
        | Beta (a, b) → (* Symbolic computation is possible *)
            ...; (* TODO *)
            bernoulli ~p:... (* TODO *)
        | _ → bernoulli ~p:(value p))
    | Gaussian (m, s) → ... (* TODO *)
    | Beta (a, b) → beta ~a:(value a) ~b:(value b)
    | V v → dirac ~v
  in (* update the score of particle prob.id *)
  prob.scores.(prob.id) ← prob.scores.(prob.id) +. logpdf marginal x
```

Plusieurs langages de programmation probabilistes exploitent les relations de conjugaisons pour améliorer la précision de l'inférence (Pyro, Augur, Birch, ProbZelus). Le moteur de calcul symbolique présenté ici est très simplifié (on ne traite que des paires de variables aléatoires conjuguées). Il est possible d'utiliser des moteurs symboliques plus puissants (e.g., *Delayed Sampling* ou *Belief Propagation*) au prix de structures de données plus élaborées.

Annexe

La sémantique d'un type $[\![t]\!]$ est l'espace mesurable de toutes les valeurs possible. Pour un environnement γ qui associe des noms de variables à des valeurs, si une expression e est de type t, la sémantique déterministe $[\![e]\!]_\gamma : t$ est une valeur de type t, et la sémantique probabiliste $\{\!|e|\!\}_\gamma : \Sigma_{[\![t]\!]} \to [0, \infty)$ est une mesure sur les valeurs de type t. On utilise la variable U pour un ensemble mesurable de valeurs. On suppose que l'environnement initial contient déjà toutes les distributions classiques (Beta, Bernoulli, Gaussian, Dirac, ...).

$$
\begin{array}{lll}
[\![c]\!]_\gamma & = c & \text{constante} \\
[\![x]\!]_\gamma & = \gamma(x) & \text{variable} \\
[\![f\ e]\!]_\gamma & = \gamma(f)\,[\![e]\!]_\gamma & \text{fonction}
\end{array}
$$

$$
\begin{array}{ll}
\{\!|\texttt{let x = } e_1 \texttt{ in } e_2|\!\}_\gamma(U) & = \int_{[\![typeOf(e1)]\!]} \{\!|e_1|\!\}_\gamma(dx) \{\!|e_2|\!\}_{\gamma[\texttt{x} \leftarrow x]}(U) \\
\{\!|\texttt{sample } e|\!\}_\gamma(U) & = [\![e]\!]_\gamma\,(U) \\
\{\!|\texttt{observe } e_1\ e_2|\!\}_\gamma(U) & = \textit{let } \mu = [\![e_1]\!]_\gamma \textit{ in } \mu_{\text{pdf}}([\![e_2]\!]_\gamma) * \delta_{()}(U)
\end{array}
$$

$$
[\![\texttt{infer } e]\!]_\gamma \qquad = \textit{let } \mu = \{\!|e|\!\}_\gamma \textit{ in } \mu(U)/\mu([\![typeOf(e)]\!])
$$

Quelques densités classiques :

$$Bernoulli(p)_{\text{pdf}}(x) = p^x(1-p)^{1-x} \qquad \text{pour } x \in \{0,1\}$$

$$Beta(a,b)_{\text{pdf}}(x) = \frac{x^{a-1}(1-x)^{b-1}}{B(a,b)} \qquad \text{pour } x \in [0,1] \text{ avec } B \text{ la fonction beta}$$

$$\mathcal{N}(\mu,\sigma)_{\text{pdf}}(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right) \qquad \text{pour } x \in \mathbb{R}$$

```
(* OCaml references *)

let x = ref 0    (* create a ref with value 0 *)
assert (!x = 0)  (* ok. use ! to access the content of a ref *)
x := 42          (* set a ref to a new value 42 *)
assert (!x = 42) (* ok. check new content *)
```

Exercice 2 : Couplage sûr - 10 points

On cherche à prouver la stabilité de l'algorithme de descente de gradient stochastique. Pour cela, on va utiliser un système de preuve basé sur HOL (Logique d'ordre supérieure), les couplages et la monade probabiliste (que l'on notera Pr). Cet exercice est tiré de l'article [2].

A. Syntaxe et sémantique du langage λ^{RP}

Grammaire des types, du langage et de la logique d'ordre supérieure (HOL)

Types	$t ::= \text{bool} \mid \text{nat} \mid \text{real} \mid \text{list}\ t \mid t \to t \mid \text{Pr}\ t$
Constantes	$c ::= \texttt{true} \mid \texttt{false} \mid n \in \mathbb{N} \mid a \in \mathbb{R}^+ \mid \mid \texttt{nil} \mid \texttt{cons} \mid$
Opérateurs	$\text{op} ::= \mid + \mid - \mid * \mid / \mid = \mid < \mid \wedge \mid \vee \mid \Rightarrow \mid \neg$
Expressions	$e ::= c \mid x \mid \text{op}(e) \mid e\ e \mid \lambda x.e \mid (e, e) \mid \text{let rec}\ f\ x = e$
	$\mid \text{if}\ e\ \text{then}\ e\ \text{else}\ e \mid \text{let}\ x = e\ \text{in}\ e$
	$\mid \text{case}\ e\ \text{of}\ \{\text{nil}\ \to\ e\ ;\ \text{cons}\ x\ x\ \to\ e\}$
	$\mid \text{dirac}\ e \mid \text{let}\ x = \text{sample}(e)\ \text{in}\ e$
	$\mid \text{bern}\ e \mid \text{choice}\ e\ e\ e \mid \text{unifL}\ e$
Prédicats	$\text{p} ::= e \mid p \wedge p \mid p \vee p \mid p \Rightarrow p \mid \neg p \mid \diamond p$
	$\mid \forall x : t.p \mid \exists x : t.p$
Environnements	$\Gamma ::= \emptyset \mid x : t, \Gamma$
Axiomes	$\Phi ::= \emptyset \mid p, \Phi$

Types avec rafinement. Le système de preuves HOL et de types raffinés (UHOL) tirés de l'article [1] dans le cas discret sont donnés en annexe. Le jugement $\Gamma; \Phi \vdash_{\text{HOL}} p$ signifie que le prédicat p prend ses variables libres dans l'environnement Γ et peut être déduit des axiomes contenus dans Φ. Le jugement de typage avec rafinement $\Gamma; \Phi \vdash e : t \mid p$ signifie que l'expression e qui a le type t dans le contexte Γ satisfait l'assertion p sous les hypothèses Φ. Le prédicat p est typé par $\Gamma, r : t \vdash p : \mathit{bool}$ et peut donc faire référence à la variable r qui représente l'expression e. Notés que les règles VAR, NIL, CONS, PAIR, SUB du système UHOL utilisent comme prémisse le système HOL. Quand Γ ou Φ sont vides ou p est `true`, on peut les omettre.
Afin de manipuler des programmes probabilistes, on ajoute aux règles décrites en annexe les règles suivantes:

$$\frac{\Gamma; \Phi \vdash e : \text{real} \mid 0 \leqslant r \leqslant 1}{\Gamma; \Phi \vdash \text{bern}\ e : \text{Pr bool}} \qquad \frac{\Gamma; \Phi \vdash e_1 : \text{bool} \qquad \Gamma; \Phi \vdash e_2 : t \mid p \qquad \Gamma; \Phi \vdash e_3 : t \mid p}{\Gamma; \Phi \vdash \text{if}\ e_1\ \text{then}\ e_2\ \text{else}\ e3 : t}$$

$$\frac{\Gamma; \Phi \vdash e : t \mid p}{\Gamma; \Phi \vdash \text{dirac}\ e : \text{Pr}\ t \mid \diamond p} \qquad \frac{\Gamma; \Phi \vdash e_1 : \text{Pr}\ t_1 \mid \diamond q \qquad \Gamma, x : t_1; \Phi, q(x) \vdash e_2 : \text{Pr}\ t_2 \mid p}{\Gamma; \Phi \vdash \text{let}\ x = \text{sample}(e_1)\ \text{in}\ e_2 : \text{Pr}\ t_2 \mid \diamond p}$$

Sémantique. La sémantique opérationnelle est donnée par un système de transitions étiquettées rappelé en annexe et on considère la sémantique dénotationnelle ensembliste qui est également rappelée en annexe. On généralise la règle de conversion du système HOL:

$$\frac{\text{CONV} \quad \Gamma \vdash e : t \qquad \Gamma \vdash e' : t' \qquad e \xrightarrow{1} e'}{\Gamma; \Phi \vdash_{\text{HOL}} e = e'}$$

Question 1. [cours] On rappelle que la correction de la sémantique vient de

$$[\![e]\!] = \sum_{e \xrightarrow{p} e'} p\,[\![e']\!]$$

Expliquer le schéma de la preuve et prouver le cas de let $x = \text{sample}(e_1)$ in e_2 dans le modèle ensembliste.

Le jugement $\Gamma; \Phi \vdash e : t \mid p$ est valide lorsque:

– pour tout q dans Φ, $\Gamma \vdash q$: bool, $\Gamma \vdash p$: bool

– pour tout environnement $\gamma \in [\![\Gamma]\!]$, Si pour tout q dans Φ, $[\![q]\!]_\gamma = \mathtt{T}$, alors $[\![p]\!]_\gamma\,([\![e]\!]_\gamma) = \mathtt{T}$ (on rappelle que T désigne le symbole "vrai" dans l'interprétation des booléens).

De la même façon, un axiome $\Gamma \vdash p$: bool est valide lorsque pour tout environnement $\gamma \in [\![\Gamma]\!]$, $[\![p]\!]_\gamma = \mathtt{T}$. Si $\Gamma, r : t \vdash p$: bool est un prédicat sur les termes de type t, alors le prédicat $\Gamma, r : \text{Pr}\ t \vdash \diamond p$: bool signifie que p est vérifié pour tous les éléments du support de la distribution:

$$[\![\diamond p]\!]_{\gamma, r \mapsto \mu} = \bigwedge_{\{r \mid \mu(r) \neq 0\}} [\![p]\!]\,(r)$$

Question 2. Justifier dans les axiomes de la forme $e = e'$ `leftId` et `assoc` $[\![e]\!] = [\![e']\!]$ (dans le modèle ensembliste).

[`leftId`] $\forall x : t\ \forall \Gamma, a : t \vdash f : \text{Pr}\ t'$

$\Gamma \vdash$ [let a = sample (Dirac x) in f] $= (\lambda a.f)\ x$

[`assoc`] $\forall \Gamma \vdash e : \text{Pr}\ t_1\ \forall \Gamma, a : t_1 \vdash f : \text{Pr}\ t_2\ \forall \Gamma, b : t_2 \vdash g : \text{Pr}\ t_3$

$\Gamma \vdash$ let y = sample (let x = sample(e) in f) in g $=$ let x = sample(e) in [let y = sample ((λa. f) x) in g]

Sucre syntaxique. Les constructions choice $p\ e_1\ e_2$ et unifL e sont du sucre syntaxique et peuvent être définies à partir des autres constructions.
La construction choice $p\ e_1\ e_2$ prend en entrée deux expressions de même type et renvoie avec probabilité p la première et avec probabilité $1 - p$ la seconde.
La construction unifL e prend une liste de type t non vide dont tous les éléments sont disjoints deux à deux et produit une distribution uniforme sur les éléments de la liste.

Question 3. Compléter les règles de typage suivantes (pour unifL e vous devrez définir le prédicat p exprimant les conditions qui peuvent faire référence à la variable libre r pour représenter e).

$$\frac{\Gamma; \Phi \vdash x : \quad | \qquad \Gamma; \Phi \vdash e_1 : \qquad \Gamma; \Phi \vdash e_2 :}{\Gamma; \Phi \vdash \text{choice}\ x\ e_1\ e_2 : \text{Pr}\ t} \qquad \frac{\Gamma; \Phi \vdash e : \text{list}\ t \mid p}{\Gamma; \Phi \vdash \text{unifL}\ e : \text{Pr}\ t}$$

Question 4. Donner des programmes définissant choice $e_1\ e_2\ e_3$ et unifL e.

Question 5. Définir la sémantique opérationnelle de choice $e_1\ e_2\ e_3$ et unifL $[a_1, \ldots, a_n]$. *Question 6.*

Définir la sémantique dénotationnelle pour $\Gamma \vdash$ choice $e_1\ e_2\ e_3$: Pr t et $\Gamma \vdash$ unifL e : Pr t.

Question 7. En prenant $\Gamma \vdash e_l$: Pr t, $\Gamma \vdash e_r$: Pr t et $\Gamma, x : t \vdash f : t'$ et comme hypothèses $\Phi = 0 < p < 1$, `leftId`, `assoc` et en utilisant le système de preuve UHOL donné en annexe, prouver le jugement

`choiceBind`

$\Gamma; \Phi \vdash$ let y = sample(choice $p\ e_l\ e_r$) in $g\ =$ choice p [let y= sample(e_l) in g] [let x= sample(e_r) in g]

B. Descente de gradient stochastique

Le but de cet exercice est de prouver la stabilité de l'algorithme de descente de gradient stochastique. Voici une implémentation d'une variante de cet algorithme:

```
val grad: (real → real) → real → real

let update z a f w = w - a * (grad (f z) w)

let rec sgd zs w0 a_s f =
  case a_s of {
  Nil → dirac w0;
  cons a a_s' →
    let z = sample (unifL ~s:zs) in
    let w = sample(sgd zs w0 a_s' f) in
    let w' = update z a f w in
    dirac w'
  }
```

L'algorithme prend en entrée:

- un poids initial w_0,
- une fonction de perte $f : Z \times \mathbb{R}^d \to \mathbb{R}$ qui est supposée L-Lipschitz, convexe et à dérivée bornée. (Nous donnerons des axiomes sur `update` qui découlent de ces propriétés, mais nous ne les manipulerons pas directement.) Z est ensemble des données d'entrainement, ici des couples de réels),
- une liste zs de tests dans Z
- une liste αs de réels, les tailles de pas d'apprentissage.

Le programme `grad` est une implémentation efficace du calcul de gradient de f dont la définition mathématique est rappelée ci-dessous où f z : $\mathbb{R}^d \to \mathbb{R}$ dont le gradient est un vecteur de longueur d:

$$\text{grad (f z) w} = \left(\frac{\partial \text{f z}}{\partial w_i}(\text{w})\right)_{1 \leqslant i \leqslant d}$$

On considère le problème de régression linéaire suivant. On veut trouver la droite $G(x) = w_1 * x + w_2$, qui représente le plus fidèlement l'ensemble de points observés $zs = [(1, 2), (2, 5), (4, 6)]$. On utilise la fonction d'objectif correspondant à la méthode des moindre carrés:

$$f : (x, y), (w_1, w_2) \mapsto (y - (w_1 * x + w_2))^2$$

avec la liste de pas $\alpha s = [0.1, 0.05, 0.1]$ et le poids initial w_0 donné par le couple $(w_{01}, w_{02}) = (1, 1)$,

Question 8. Exécuter l'algorithme `sgd` sur ces données en détaillant les différentes étapes succinctement.

Pour la suite, on considérera que $d = 1$ par simplicité, c'est-à-dire que l'on a un seul paramètre w à optimiser.

C. Distances et couplage

Rappelons qu'une distance est une fonction $d : t \times t \to \mathbb{R}^+$ qui vérifie les axiomes

$$\begin{aligned}
\texttt{IdDist} \quad & \forall x, y : t \quad d_t(x, y) = 0 \iff x = y \\
\texttt{SymDist} \quad & \forall x, y : t \quad d_t(x, y) = d_t(y, x) \\
\texttt{TrDist} \quad & \forall x, y, z : t \quad d(x, z) \leqslant d(x, y) + d(y, z)
\end{aligned}$$

Pour étudier la stabilité de l'algorithme, on va introduire des distances pour chaque type:

$$
\begin{aligned}
&d_{\text{bool}}(b_1, b_2) = \text{if } b_1 = b_2 \text{ then } 0 \text{ else } 1\\
&d_{\text{nat}}(n_1, n_2) = |n_1 - n_2|\\
&d_{\text{real}}(a_1, a_2) = |a_1 - a_2|\\
&d_{\text{list } t}(l_1, l_2) = \text{if len}(l_1) \neq \text{len}(l_2) \text{ then } \infty \text{ else } \max\{d_t(x_i, y_i) \mid 1 \leqslant i \leqslant \text{len}(x)\}\\
&d_{t \to t'}(f_1, f_2) = \max\{d_{t'}(f_1(x), f_2(x)) \mid x : t\}
\end{aligned}
$$

Il reste à définir la distance entre les distributions de probabilité et pour ça on introduit les couplages et la distance de Kantorowitch.
Étant donnée une distribution μ sur $C_1 \times C_2$, les premières et secondes **marginales** de μ sont des distributions sur C_1 et C_2 respectivement définies par:

$$\pi_1(\mu)(x) = \sum_{y \in C_2} \mu(x, y) \qquad \pi_2(\mu)(y) = \sum_{x \in C_1} \mu(x, y)$$

Un **couplage** de deux distributions μ_1 et μ_2 est une distribution dont les marginales sont μ_1 et μ_2 respectivement.

Question 9. Définir un couplage pour les distributions suivantes:

$$
\begin{aligned}
&(\text{bern } p, \text{bern } p'))\\
&(\text{bern } p, \text{unifL } [5, 2, 1]))\\
&(\text{unifL } [i_1, \ldots, i_k], \text{unifL } [j_1, \ldots, j_\ell]))
\end{aligned}
$$

La distance de Kantorowitch entre deux distributions est la plus petite espérance des distances entre les échantillons, pondérée par un couplage:

$$d_{\Pr t}(\mu_1, \mu_2) = \inf \left\{ \sum_{x,y} \mu(x, y) d_t(x, y) \mid \mu \in \Pr(t_1 \times t_2),\ \pi_1(\mu) = \mu_1, \pi_2(\mu) = \mu_2 \right\}$$

Question 10. Montrer les énoncés suivant en majorant la distance de Kantorowitch entre les distributions à l'aide de couplages bien choisis.

$$
\begin{aligned}
&d_{\Pr \text{ bool}}(\text{bern } p, \text{bern } p')) \leqslant |p - p'|\\
&d_{\Pr \text{ nat}}(\text{unifL } [1, 2], \text{unifL } [3, 1])) = 1/2\\
&d_{\Pr \text{ nat}}(\text{unifL } [1, 2, 4], \text{unifL } [4, 2, 1])) = 0
\end{aligned}
$$

Question 11. En supposant que d_t est une distance, montrer que $d_{\Pr t}$ est une distance.

Question 12. Montrer les propriétés suivantes en raisonnant sur les couplages possibles:

`pureDist` $\forall t\ \forall x_l, x_r : \text{t}$
$d_{\Pr \text{t}}(\text{dirac } x_l, \text{dirac } x_r) = d_{\text{t}}(x_l, x_r)$

`pureBindDist` $\forall t\ \forall e_l, e_r : \Pr \text{t}\ \forall f_l, f_r : t \to t'$
$\forall x_l, x_r : t\ d_{t'}(f_l\ x_l, f_r\ x_r) - d_t(x_l, x_r) \leqslant m \Rightarrow$
$d_{\Pr \text{t'}}(\text{let x = sample } e_l \text{ in dirac } (f_l\ x), \text{let x = sample } e_r \text{ in dirac } (f_r\ x)) \leqslant d_{\Pr \text{t}}(e_l, e_r) + m$

Question 13. Après avoir défini le prédicat isPermutation, montrer les propriétés suivantes:

`unifDist` $\forall t\ \forall xs_l, xs_r$: list t
isPermutation $xs_l\ xs_r \Rightarrow d_{\text{Pr t}}(\text{unifL}\ xs_l, \text{unifL}\ xs_r) = 0$

`choiceDist` $\forall p$: real $\forall t\ \forall e_l, e'_l, e_r, e'_r$: Pr t
$d_{\text{Pr t}}(\text{choice}\ p\ e_l\ e'_l, \text{choice}\ p\ e_r\ e'_r) \leqslant p * d_{\text{Pr}\,t}(e_l, e_r) + (1-p) * d_{\text{Pr}\,t}(e'_l, e'_r)$

Couplage de Kantorowitch. Étant donnés:

– des distributions μ_1 sur C_1 et μ_2 sur C_2

– une relation $R \subseteq C_1 \times C_2$

– une distance $d : C_1 \times C_2 \to \mathbb{R}^+$

– une borne $k \in \mathbb{R}^+$

On dit que leur **couplage de Kantorowitch** est vérifié, et on note $\models \diamond_k R(\mu_1, \mu_2)$ si et seulement si il existe une distribution μ sur $C_1 \times C_2$ telle que les conditions suivantes sont vérifiées:

Marginales: $\mu_1 = \pi_1(\mu)$ et $\mu_2 = \pi_2(\mu)$

Couplage: supp $\mu = \{(x, y) \mid \mu(x, y) \neq 0\} \subseteq R$

Kantorowitch: $\displaystyle\sum_{(x,y)\in C_1\times C_2} \mu(x, y)d(x, y) \leqslant k$

Système de type avec couplage. un jugement $\Gamma; \Phi \vdash e_l : t_l \sim e_r : t_r \mid R$ englobe le typage $\Gamma \vdash e_l : t_l$ et $\Gamma \vdash e_r : t_r$ et la preuve que les expressions e_l et e_r satisfont la relation R sous les hypothèses Φ (dont les variables libres sont dans Γ).

T-BOT

$$\frac{}{\Gamma; \Phi \vdash e_l : \text{Pr}\ t_l \ \sim\ e_r : \text{Pr}\ t_r \mid \diamond_\infty \texttt{true}}$$

T-WEAKEN

$$\frac{\Gamma, r_l : t_l, r_r : t_r; \Phi \vdash p \Rightarrow p_w \qquad \Gamma, r_l : t_l, r_r : t_r; \Phi \vdash k \leqslant k_w \qquad \Gamma; \Phi \vdash e_l : \text{Pr}\ t_l \ \sim\ e_r : \text{Pr}\ t_r \mid \diamond_k p}{\Gamma; \Phi \vdash e_l : \text{Pr}\ t_l \ \sim\ e_r : \text{Pr}\ t_r \mid \diamond_{k_w} p_w}$$

T-BERN

$$\frac{\Gamma; \Phi \vdash e_l : \text{real} \ \sim\ e_r : \text{real} \mid 0 \leqslant r_l \leqslant r_r \leqslant 1}{\Gamma; \Phi \vdash \text{bern}\ e_l : \text{Pr bool} \ \sim\ \text{bern}\ e_r : \text{Pr bool} \mid \diamond_{|r_r - r_l|} \texttt{true}}$$

T-DIRAC

$$\frac{\Gamma; \Phi \vdash e_l : \text{Pr}\ t \ \sim\ e_r : \text{Pr}\ t \mid p\ r_l\ r_r \wedge d_t(r_l, r_r) \leqslant k}{\Gamma; \Phi \vdash \text{dirac}\ e_l : \text{Pr}\ t \ \sim\ \text{dirac}\ e_r : \text{Pr}\ t \mid \diamond_k p}$$

T-SAMPLE

$$\frac{\Gamma; \Phi \vdash e_l : \text{Pr}\ s \ \sim\ e_r : \text{Pr}\ s \mid \diamond_k p \qquad \Gamma, x_l : s, x_r : s; \Phi, p\ x_l\ x_r \vdash f_l\ x_l : \text{Pr}\ t \ \sim\ f_r\ x_r : \text{Pr}\ t \mid \diamond_{a*d_s(r_l,r_r)+b} q}{\Gamma; \Phi \vdash \text{let x=sample}(e_l)\ \text{in}\ f_l\ \text{x} : \text{Pr}\ t_l \ \sim\ \text{let x=sample}(e_r)\ \text{in}\ f_r\ \text{x} : \text{Pr}\ t \mid \diamond_{a*k+b} q}$$

FIGURE 1 – Système de type relationnel probabiliste (Sélection de règles)

La règle T-Bot signifie que quelque soit les termes e_l et e_r représentant des distributions de probabilités, la distance de Kantorowitch entre les deux distributions est bornée.

Question 14. Après avoir donné leur interprétation, prouver la correction des règles `T-Bern`, et `T-Dirac`. On admettra la correction de la règle `T-Sample` qui se ramène aux propriétés de `pureBindDist`

Question 15. Donner les règles `T-UnifL` et `T-Choice`.

Question 16. Énoncer le théorème de correction du système de preuve relationnel probabiliste. Donner le schéma de la preuve de correction.

D. Stabilité de SGD.

On souhaite prouver la stabilité de l'algorithme sgd, qui assure que la distance de Kantorowitch entre deux exécutions de l'algorithme SGD sur deux ensembles de données zs_l et zs_r qui différent exactement d'un élément est bornée par la distance entre les poids initiaux plus un ϵ qui ne dépend pas des données sur lesquelles les ensembles de données diffèrent.
Ainsi, la stabilité est exprimée par: pour toutes données x, y et liste de données zs tels que $x :: zs$ et $y :: zs$ sont des permutations respectives de zs_l et zs_r, pour tous $w0_l$, $w0_r$, αs, f, il existe L tel que

$$d_{\text{Pr real}}(\text{sgd } zs_l\ w0_l\ \alpha s\ f,\ \text{sgd } zs_r\ w0_r\ \alpha s\ f) \leqslant d_{\text{real}}(w0_l,\ w0_r) + 2 * L * \frac{\text{sum}(\alpha s)}{\text{len}(zs) + 1} \tag{1}$$

On va admettre deux propriétés sur `update` que l'on peut prouver à partir des propriétés de f (L-lipschitzienne, convexe et à dérivée bornée):

$$\begin{aligned} \texttt{contractive} \quad & d_{\text{real}}(\text{update } z\ \alpha\ f\ w_l, \text{update } z\ \alpha\ f\ w_r) < d_{\text{real}}(w_l, w_r) \\ \texttt{bounded} \quad & d_{\text{real}}(\text{update } z_l\ \alpha\ f\ w_l, \text{update } z_r\ \alpha\ f\ w_r) < d_{\text{real}}(w_l, w_r) + 2L\alpha \end{aligned}$$

On raisonne par induction sur le nombre de pas $\text{len}(\alpha s)$.

Question 17. Montrer que la propriété (1) est vérifiée lorsque $\alpha s = \text{nil}$. *Question 18.* Pour le cas inductif,

réécrire `sgd` en utilisant la définition de `unifL` que vous avez donnée en fonction de `choice`. Puis utiliser l'axiome `choiceBind` pour simplifier l'expression.

Pour calculer la borne sur la distance entre les deux instances de `sgd` on raisonne par cas sur les échantillons et on utilise les axiomes `choiceDist` et `pureBindDist` et l'hypothèse d'induction:

Question 19. dans le cas où les deux échantillons sont les mêmes, majorer la distance en utilisant les axiomes `leftId` et `contractive`.

Question 20. dans le cas où les deux échantillons sont différents, majorer la distance en utilisant l'axiome `bounded`.

Annexes

Les figures représentant les systèmes de types et de preuve suivant sont tirées de l'article [1].

Système de preuve HOL (sélection de règles)

$$\frac{\phi \in \Psi}{\Gamma \mid \Psi \vdash \phi}\ \mathsf{AX} \qquad \frac{\Gamma \vdash t : \tau \quad \Gamma \vdash t' : \tau \quad t =_{\beta\iota\mu} t'}{\Gamma \mid \Psi \vdash t = t'}\ \mathsf{CONV} \qquad \frac{\Gamma \mid \Psi \vdash \phi[t/x] \quad \Gamma \mid \Psi \vdash t = u}{\Gamma \mid \Psi \vdash \phi[u/x]}\ \mathsf{SUBST}$$

$$\frac{\Gamma \mid \Psi, \psi \vdash \phi}{\Gamma \mid \Psi \vdash \psi \Rightarrow \phi}\ \Rightarrow_{\mathsf{I}} \qquad \frac{\Gamma \mid \Psi \vdash \psi \Rightarrow \phi \quad \Gamma \mid \Psi \vdash \psi}{\Gamma \mid \Psi \vdash \phi}\ \Rightarrow_{\mathsf{E}} \qquad \frac{\Gamma, x : \sigma \mid \Psi \vdash \phi}{\Gamma \mid \Psi \vdash \forall x : \sigma.\phi}\ \forall_{\mathsf{I}}$$

$$\frac{\Gamma \mid \Psi \vdash \forall x : \sigma.\phi \quad \Gamma \vdash t : \sigma}{\Gamma \mid \Psi \vdash \phi[t/x]}\ \forall_{\mathsf{E}} \qquad \frac{}{\Gamma \mid \Psi \vdash \top}\ \top_{\mathsf{I}} \qquad \frac{\Gamma \mid \Psi \vdash \bot \quad \Gamma \vdash \phi}{\Gamma \mid \Psi \vdash \phi}\ \bot_{\mathsf{E}}$$

$$\frac{\Gamma \mid \Psi \vdash \phi[[]/l] \quad \Gamma, h : \tau, t : \mathsf{list}_\tau \mid \Psi, \phi \vdash \phi[h :: t/t]}{\Gamma \mid \Psi \vdash \forall t : \mathsf{list}_\sigma.\phi}\ \mathsf{LIST} \qquad \frac{\Gamma \vdash h :: t : \mathsf{list}_\tau}{\Gamma \mid \emptyset \vdash [] \neq h :: t}\ \mathsf{NC}$$

$$\frac{\Gamma \mid \Psi \vdash t_1 :: t_2 = t_1' :: t_2'}{\Gamma \mid \Psi \vdash t_i = t_i'}\ \mathsf{CONS_i} \qquad \frac{\Gamma, t : \mathsf{list}_\tau \mid \Psi, \forall u : \mathsf{list}_\tau.|u| < |t| \Rightarrow \phi[u/t] \vdash \phi}{\Gamma \mid \Psi \vdash \forall t : \mathsf{list}_\tau.\phi}\ \mathsf{SLIST}$$

Système de typage UHOL (sélection de règles)

$$\frac{\Gamma \vdash x : \sigma \qquad \Gamma \mid \Psi \vdash \phi[x/\mathbf{r}]}{\Gamma \mid \Psi \vdash x : \sigma \mid \phi}\ \mathsf{VAR} \qquad \frac{\Gamma, x : \tau \mid \Psi, \phi' \vdash t : \sigma \mid \phi}{\Gamma \mid \Psi \vdash \lambda x : \tau.t : \tau \to \sigma \mid \forall x.\phi' \Rightarrow \phi[\mathbf{r}\ x/\mathbf{r}]}\ \mathsf{ABS}$$

$$\frac{\Gamma \mid \Psi \vdash t : \tau \to \sigma \mid \forall x.\phi'[x/\mathbf{r}] \Rightarrow \phi[\mathbf{r}\ x/\mathbf{r}] \qquad \Gamma \mid \Psi \vdash u : \tau \mid \phi'}{\Gamma \mid \Psi \vdash t\ u : \sigma \mid \phi[u/x]}\ \mathsf{APP} \qquad \frac{\Gamma \mid \Psi \vdash_{\mathsf{HOL}} \phi[[]/\mathbf{r}]}{\Gamma \mid \Psi \vdash [] : \mathsf{list}_\sigma \mid \phi}\ \mathsf{NIL}$$

$$\frac{\begin{array}{c}\Gamma \mid \Psi \vdash h : \sigma \mid \phi' \qquad \Gamma \mid \Psi \vdash t : \mathsf{list}_\sigma \mid \phi'' \\ \Gamma \mid \Psi \vdash_{\mathsf{HOL}} \forall xy.\phi'[x/\mathbf{r}] \Rightarrow \phi''[y/\mathbf{r}] \Rightarrow \phi[x :: y/\mathbf{r}]\end{array}}{\Gamma \mid \Psi \vdash h :: t : \mathsf{list}_\sigma \mid \phi}\ \mathsf{CONS} \qquad \frac{\Gamma \mid \Psi \vdash t : \sigma \times \tau \mid \phi[\pi_i(\mathbf{r})/\mathbf{r}]}{\Gamma \mid \Psi \vdash \pi_i(t) : \sigma \mid \phi}\ \mathsf{PROJ_i}$$

$$\frac{\begin{array}{c}\Gamma \mid \Psi \vdash t : \sigma \mid \phi' \qquad \Gamma \mid \Psi \vdash u : \tau \mid \phi'' \\ \Gamma \mid \Psi \vdash_{\mathsf{HOL}} \forall xy.\phi'[x/\mathbf{r}] \Rightarrow \phi''[y/\mathbf{r}] \Rightarrow \phi[\langle x, y\rangle/\mathbf{r}]\end{array}}{\Gamma \mid \Psi \vdash \langle t, u\rangle : \sigma \times \tau \mid \phi}\ \mathsf{PAIR} \qquad \frac{\Gamma \mid \Psi \vdash t : \sigma \mid \phi' \quad \Gamma \mid \Psi \vdash_{\mathsf{HOL}} \phi'[t/\mathbf{r}] \Rightarrow \phi[t/\mathbf{r}]}{\Gamma \mid \Psi \vdash t : \sigma \mid \phi}\ \mathsf{SUB}$$

$$\frac{\begin{array}{c}\Gamma \vdash l : \mathsf{list}_\tau \\ \Gamma \mid \Psi, l = [] \vdash u : \sigma \mid \phi \\ \Gamma \mid \Psi \vdash v : \tau \to \mathsf{list}_\tau \to \sigma \mid \forall ht.l = h :: t \Rightarrow \phi[\mathbf{r}\ h\ t/\mathbf{r}]\end{array}}{\Gamma \mid \Psi \vdash \mathsf{case}\ l\ \mathsf{of}\ [] \mapsto u; \_ :: \_ \mapsto v : \sigma \mid \phi}\ \mathsf{LISTCASE}$$

$$\frac{\begin{array}{c}\mathcal{D}ef(f, x, e) \\ \Gamma, x : I, f : I \to \sigma \mid \Psi, \phi', \forall m.|m| < |x| \Rightarrow \phi'[m/x] \Rightarrow \phi[m/x][f\ m/\mathbf{r}] \vdash e : \sigma \mid \phi\end{array}}{\Gamma \mid \Psi \vdash \mathsf{letrec}\ f\ x\ = e : I \to \sigma \mid \forall x.\phi' \Rightarrow \phi[\mathbf{r}\ x/\mathbf{r}]}\ \mathsf{LETREC} \qquad \text{where } I \in \{\mathbb{N}, \mathsf{list}_\tau\}$$

Système de typage RHOL (sélection de règles)

$$\frac{\Gamma, x_1 : \tau_1, x_2 : \tau_2 \mid \Psi, \phi' \vdash t_1 : \sigma_1 \sim t_2 : \sigma_2 \mid \phi}{\Gamma \mid \Psi \vdash \lambda x_1 : \tau_1.t_1 : \tau_1 \to \sigma_1 \sim \lambda x_2 : \tau_2.t_2 : \tau_2 \to \sigma_2 \mid \forall x_1, x_2.\phi' \Rightarrow \phi[\mathbf{r}_1\ x_1/\mathbf{r}_1][\mathbf{r}_2\ x_2/\mathbf{r}_2]}\ \textsf{ABS}$$

$$\frac{\begin{array}{c}\Gamma \mid \Psi \vdash t_1 : \tau_1 \to \sigma_1 \sim t_2 : \tau_2 \to \sigma_2 \mid \forall x_1, x_2.\phi'[x_1/\mathbf{r}_1][x_2/\mathbf{r}_2] \Rightarrow \phi[\mathbf{r}_1\ x_1/\mathbf{r}_1][\mathbf{r}_2\ x_2/\mathbf{r}_2] \\ \Gamma \mid \Psi \vdash u_1 : \tau_1 \sim u_2 : \tau_2 \mid \phi'\end{array}}{\Gamma \mid \Psi \vdash t_1 u_1 : \sigma_1 \sim t_2 u_2 : \sigma_2 \mid \phi[u_1/x_1][u_2/x_2]}\ \textsf{APP}$$

$$\frac{\Gamma \vdash x_1 : \sigma_1 \quad \Gamma \vdash x_2 : \sigma_2 \quad \Gamma \mid \Psi \vdash \phi[x_1/\mathbf{r}_1][x_2/\mathbf{r}_2]}{\Gamma \mid \Psi \vdash x_1 : \sigma_1 \sim x_2 : \sigma_2 \mid \phi}\ \textsf{VAR} \qquad \frac{\Gamma \mid \Psi \vdash_{\mathsf{HOL}} \phi[\mathtt{tt}/\mathbf{r}_1][\mathtt{tt}/\mathbf{r}_2]}{\Gamma \mid \Psi \vdash \mathtt{tt} : \mathbb{B} \sim \mathtt{tt} : \mathbb{B} \mid \phi}\ \textsf{TRUE}$$

$$\frac{\Gamma \mid \Psi \vdash_{\mathsf{HOL}} \phi[[]/\mathbf{r}_1][[]/\mathbf{r}_2]}{\Gamma \mid \Psi \vdash [] : \mathsf{list}_{\sigma_1} \sim [] : \mathsf{list}_{\sigma_2} \mid \phi}\ \textsf{NIL}$$

$$\frac{\begin{array}{c}\Gamma \mid \Psi \vdash h_1 : \sigma_1 \sim h_2 : \sigma_2 \mid \phi' \qquad \Gamma \mid \Psi \vdash t_1 : \mathsf{list}_{\sigma_1} \sim t_2 : \mathsf{list}_{\sigma_2} \mid \phi'' \\ \Gamma \mid \Psi \vdash_{\mathsf{HOL}} \forall x_1 x_2 y_1 y_2.\phi'[x_1/\mathbf{r}_1][x_2/\mathbf{r}_2] \Rightarrow \phi''[y_1/\mathbf{r}_1][y_2/\mathbf{r}_2] \Rightarrow \phi[x_1 :: y_1/\mathbf{r}_1][x_2 :: y_2/\mathbf{r}_2]\end{array}}{\Gamma \mid \Psi \vdash h_1 :: t_1 : \mathsf{list}_{\sigma_1} \sim h_2 :: t_2 : \mathsf{list}_{\sigma_2} \mid \phi}\ \textsf{CONS}$$

$$\frac{\begin{array}{c}\Gamma \mid \Psi \vdash l_1 : \mathsf{list}_{\tau_1} \sim l_2 : \mathsf{list}_{\tau_2} \mid \mathbf{r}_1 = [] \Leftrightarrow \mathbf{r}_2 = [] \\ \Gamma \mid \Psi, l_1 = [], l_2 = [] \vdash u_1 : \sigma_1 \sim u_2 : \sigma_2 \mid \phi \\ \Gamma \mid \Psi \vdash v_1 : \tau_1 \to \mathsf{list}_{\tau_1} \to \sigma_1 \sim v_2 : \tau_2 \to \mathsf{list}_{\tau_2} \to \sigma_2 \mid \\ \forall h_1 h_2 t_1 t_2.l_1 = h_1 :: t_1 \Rightarrow l_2 = h_2 :: t_2 \Rightarrow \phi[\mathbf{r}_1\ h_1\ t_1/\mathbf{r}_1][\mathbf{r}_2\ h_2\ t_2/\mathbf{r}_2]\end{array}}{\Gamma \mid \Psi \vdash \mathsf{case}\ l_1\ \mathsf{of}\ [] \mapsto u_1; \_ :: \_ \mapsto v_1 : \sigma_1 \sim \mathsf{case}\ l_2\ \mathsf{of}\ [] \mapsto u_2; \_ :: \_ \mapsto v_2 : \sigma_2 \mid \phi}\ \textsf{LISTCASE}$$

$$\frac{\begin{array}{c}\Gamma \mid \Psi \vdash t_1 : \sigma_1 \sim t_2 : \sigma_2 \mid \phi' \qquad \Gamma \mid \Psi \vdash u_1 : \tau_1 \sim u_2 : \tau_2 \mid \phi'' \\ \Gamma \mid \Psi \vdash_{\mathsf{HOL}} \forall x_1 x_2 y_1 y_2.\phi'[x_1/\mathbf{r}_1][x_2/\mathbf{r}_2] \Rightarrow \phi''[y_1/\mathbf{r}_1][y_2/\mathbf{r}_2] \Rightarrow \phi[\langle x_1, y_1\rangle/\mathbf{r}_1][\langle x_2, y_2\rangle/\mathbf{r}_2]\end{array}}{\Gamma \mid \Psi \vdash \langle t_1, u_1\rangle : \sigma_1 \times \tau_1 \sim \langle t_2, u_2\rangle : \sigma_2 \times \tau_2 \mid \phi}\ \textsf{PAIR}$$

$$\frac{\Gamma \mid \Psi \vdash t_1 : \sigma_1 \times \tau_1 \sim t_2 : \sigma_2 \times \tau_2 \mid \phi[\pi_i(\mathbf{r}_1)/\mathbf{r}_1][\pi_i(\mathbf{r}_2)/\mathbf{r}_2]}{\Gamma \mid \Psi \vdash \pi_i(t_1) : \sigma_1 \sim \pi_i(t_2) : \sigma_2 \mid \phi}\ \textsf{PROJ}_\textsf{i}$$

$$\frac{\Gamma \mid \Psi \vdash t_1 : \sigma_1 \sim t_2 : \sigma_2 \mid \phi' \quad \Gamma \mid \Psi \vdash_{\mathsf{HOL}} \phi'[t_1/\mathbf{r}_1][t_2/\mathbf{r}_2] \Rightarrow \phi[t_1/\mathbf{r}_1][t_2/\mathbf{r}_2]}{\Gamma \mid \Psi \vdash t_1 : \sigma_1 \sim t_2 : \sigma_2 \mid \phi}\ \textsf{SUB}$$

$$\frac{\Gamma \mid \Psi \vdash t_1 : \sigma_2 \sim t_2 : \sigma_2 \mid \phi \quad \Gamma \mid \Psi \vdash t_1 : \sigma_2 \sim t_2 : \sigma_2 \mid \phi'}{\Gamma \mid \Psi \vdash t_1 : \sigma_2 \sim t_2 : \sigma_2 \mid \phi \wedge \phi'}\ \wedge_\textsf{I} \qquad \frac{\Gamma \mid \Psi, \phi'[t_1/\mathbf{r}_1][t_2/\mathbf{r}_2] \vdash t_1 : \sigma_2 \sim t_2 : \sigma_2 \mid \phi}{\Gamma \mid \Psi \vdash t_1 : \sigma_2 \sim t_2 : \sigma_2 \mid \phi' \Rightarrow \phi}\ \Rightarrow_\textsf{I}$$

$$\frac{\Gamma \mid \Psi \vdash t_1 : \sigma_1 \mid \phi[\mathbf{r}/\mathbf{r}_1][t_2/\mathbf{r}_2]}{\Gamma \mid \Psi \vdash t_1 : \sigma_1 \sim t_2 : \sigma_1 \mid \phi}\ \textsf{UHOL} - \textsf{L}$$

Sémantique opérationnelle. La sémantique opérationnelle est donnée par un système de transtions étiquettées par la probabiité (discrète) de réduire:

$$\frac{}{\text{sample}(\text{bern } p) \xrightarrow{p} \texttt{true}} \qquad \frac{}{\text{sample}(\text{bern } p) \xrightarrow{1-p} \texttt{false}} \qquad \frac{e \xrightarrow{r} e'}{\text{bern } e \xrightarrow{r} \text{bern } e'}$$

$$\frac{}{\text{sample}(\text{dirac } e) \xrightarrow{1} e} \qquad \frac{}{\text{let } x = v \text{ in } e_2 \xrightarrow{1} e_2[v/x]} \qquad \frac{e_1 \xrightarrow{r} e_1'}{\text{let } x = e_1 \text{ in } e_2 \xrightarrow{r} \text{let } x = e_1' \text{ in } e_2}$$

$$\frac{}{(\lambda x.e_2)e_1 \xrightarrow{1} e_2[e_1/x]} \qquad \frac{e_2 \xrightarrow{r} e_2'}{(e_2)e_1 \xrightarrow{r} (e_2')e_1} \qquad \frac{}{\text{let rec } f\ x = e \xrightarrow{1} e[\text{let rec } f\ x = e/x]}$$

$$\frac{}{\text{if true then } e_2 \text{ else } e_3 \xrightarrow{1} e_2} \qquad \frac{}{\text{if false then } e_2 \text{ else } e_3 \xrightarrow{1} e_3} \qquad \frac{e_1 \xrightarrow{r} e_1'}{\text{if } e_1 \text{ then } e_2 \text{ else } e_3 \xrightarrow{r} \text{if } e_1' \text{ then } e_2 \text{ else } e_3}$$

$$\frac{\Gamma \vdash e_1 : \mathit{bool}}{\text{let x = sample (if}e_1 \text{ then } e_2 \text{ else } e_3) \text{ in } e_4 \xrightarrow{1} \text{if}e_1 \text{ then (let x = sample}(e_2) \text{ in } e_4) \text{ else (let x = sample}(e_3) \text{ in } e_4)}$$

Sémantique ensembliste. On note $\mathrm{D}(X)$ l'ensemble des distributions de probabilités discrètes sur l'ensemble X:

$$\mathrm{D}(X) = \left\{ \mu : X \to \mathbb{R}^+ \mid \sum_{x \in X} \mu(x) = 1 \right\}$$

On note $\delta_x : y \mapsto 1$ si $x = y; x \mapsto 0$ si $x \neq y$
On note $B_p : \mathtt{T} \mapsto p\ ;\ \mathtt{F} \mapsto 1 - p$ la distribution de Bernoulli de paramètre p.
On note $U_V : x \mapsto \frac{1}{\text{card}(V)}$ si $x \in V; x \mapsto 0$ si $x \notin V$ la distribution uniforme sur un ensemble fini V.

Les types sont interprétés par des ensembles.

$$[\![\text{bool}]\!] = \{\mathtt{T}, \mathtt{F}\} \qquad [\![\text{nat}]\!] = \mathbb{N} \qquad [\![\text{list } t]\!] = \text{list}_{[\![t]\!]}$$
$$[\![\text{real}]\!] = \mathbb{R} \qquad [\![t \to t']\!] = \text{Set}([\![t]\!], [\![t']\!]) \qquad [\![\text{Pr } t]\!] = \mathrm{D}([\![t]\!])$$

Le contexte Γ est interprété par un ensemble d'environnements:

$$[\![\Gamma]\!] = \{\gamma \mid \forall x : t \in \Gamma\ \gamma(x) \in [\![t]\!]\}$$

On donne l'interprétation ensembliste usuelle aux termes déterministes $[\![\Gamma \vdash e : t]\!]_\gamma \in [\![t]\!]$ et on rappelle l'interprétation des termes probabilistes:

$$[\![\Gamma \vdash \text{bern } e]\!]_\gamma = B_{[\![e]\!]_\gamma} \qquad [\![\Gamma \vdash \text{dirac } e]\!]_\gamma = \delta_{[\![e]\!]_\gamma}$$
$$[\![\Gamma \vdash \text{let x = sample}(e_1) \text{ in } e_2]\!]_\gamma = \sum_{a \in [\![t]\!]} [\![\Gamma \vdash e_1]\!]_\gamma (a)\, [\![\Gamma, x : t \vdash e_2]\!]_{\gamma, x \mapsto a}$$

Références

[1] Alejandro Aguirre, Gilles Barthe, Marco Gaboardi, Deepak Garg, and Pierre-Yves Strub. A relational logic for higher-order programs. *J. Funct. Program.*, 29:e16, 2019.

[2] Elizaveta Vasilenko, Niki Vazou, and Gilles Barthe. Safe couplings: coupled refinement types. *Proc. ACM Program. Lang.*, 6(ICFP):596–624, 2022.