TAS
Cours 03 - Lambda-Calcul : Polymorphisme

Romain Demangeon

TAS - M2 STL

03/10/2024

SORBONNE UNIVERSITÉ

Rappels

- Règles de sémantique de λ :

$$(\beta)\,\frac{}{\lambda x.M\ N \longrightarrow M[N/x]} \qquad (\mu_1)\,\frac{M \longrightarrow M'}{M\ N \longrightarrow M'\ N}$$

$$(\mu_2)\,\frac{N \longrightarrow N'}{M\ N \longrightarrow M\ N'} \qquad (\xi)\,\frac{M \longrightarrow M'}{\lambda x.M \longrightarrow \lambda x.M'}$$

- Règles de typage de λ_{ST} :

$$(\mathbf{Var})\,\frac{}{\Gamma, x : T \vdash x : T} \qquad (\mathbf{Abs})\,\frac{\Gamma, x : T \vdash M : S}{\Gamma \vdash \lambda x.M : T \to S}$$

$$(\mathbf{App})\,\frac{\Gamma \vdash M : S \to T \qquad \Gamma \vdash N : S}{\Gamma \vdash M\ N : T}$$

Inférence de λ_{ST} : Génération d'équations

(dans la suite t_i variable et T type)

- Jugement initial à partir du terme à typer M_0 et d'une première variable (but) T_0: $\Gamma \vdash M_0 : t_0$
- Le système de types est dirigé par la syntaxe : à chaque étape, une seule règle applicable.
- Cas $\Gamma \vdash \lambda x.M : T$
  - on crée deux nouvelles variables t_k et t_m
  - on génère les équations de $\Gamma, x : t_k \vdash M : t_m$
  - on ajoute l'équation $T = t_k \to t_m$
- Cas $\Gamma \vdash M \ N : T$
  - on crée une nouvelle variable t_k
  - on génère les équations de $\Gamma \vdash M : t_k \to T$
  - on génère les équations de $\Gamma \vdash N : t_k$
  - on fait l'union des deux ensembles d'équations.
- Cas $\Gamma \vdash x : T$
  - on cherche si $x : T'$ appartient à Γ
  - si oui on produit une unique équation $T' = T$
  - si non, on échoue (le terme n'est pas typable).
    (il nous manque une information sur une variable libre)

SORBONNE UNIVERSITÉ

Inférence de λ_{ST} : Unification

- Signature : liste de symboles de fonctions avec leur arité.
- Terme : arbre syntaxique contenant des variables et des symboles de fonctions (ou constantes)
- Terme clos: terme sans variable.
- Algèbre de termes: ensembles de tous les termes définis inductivement[1] depuis une signature et un ensemble (infini) de variables.
- Exemple
  - signature : $\Sigma_1 = \{(Z,0),(S,1)\}$
  - Z est un terme clos de Σ_1
  - x est un terme de Σ_1
  - $S(S(S(Z)))$ est un terme clos de Σ_1
  - $S(y)$ est un terme de Σ_1
- Exemple
  - signature : $\Sigma_2 = \{(Z,0),(S,1),(+,2),(.,2)\}$
  - $S(S(S(Z)))$ est un terme clos de Σ_2
  - $+(.(Z,S(Z)),S(S(S(Z))))$ est un terme clos de Σ_2
  - $+(.(y,S(Z)),S(S(x)))$ est un terme de Σ_2

[1] plus petit point fixe de Knaster-Tarski.

Inférence de λ_{ST} : Unification (II)

- Exemple
  - signature : $\Sigma_3 = \{(\rightarrow, 2)\}$
  - $t_1 \rightarrow (t_2 \rightarrow t_1)$ est un terme de Σ_3.
  - il n'existe pas de termes clos sur Σ_3.

Inférence de λ_{ST} : Unification (II)

- Exemple
  - signature : $\Sigma_3 = \{(\rightarrow, 2)\}$
  - $t_1 \rightarrow (t_2 \rightarrow t_1)$ est un terme de Σ_3.
  - il n'existe pas de termes clos sur Σ_3.
  - Σ_3 est la signature des types de λ_{ST}.
- Exemple:
  - on veut définir les termes du λ-calcul pur,
  - on génère les termes de la signature $\{(\mathtt{App}, 2), (\lambda, 2)\}$
  - on se restreint aux termes tels que, pour tout noeud $\lambda(M_1, M_2)$, M_1 est une variable.
- une équation est un couple de termes (sur la même signature)
  - par exemple : $S_1 = (x + (y.SSZ),\ Z + v)$
  - par exemple : $S_2 = (t_1 \rightarrow t_2, (t_2 \rightarrow t_2) \rightarrow t_3)$

Inférence de λ_{ST} : Unification (III)

- un système est un ensemble d'équations :
  - par exemple : $\{(y, S(x)), (x + (y.SSZ),\ Z + v)\}$
  - par exemple : $\{(t_1 \to t_2, (t_2 \to t_2) \to t_3)\}$
- un substitution est une fonction des variables dans les termes qui est l'identité presque partout.
  - par exemple : σ_1 définie par $x \mapsto SSZ, y \mapsto Z$ (et l'identité ailleurs)
  - par exemple : σ_2 définie par $x \mapsto Z, v \mapsto (SZ.SSZ), y \mapsto SZ$
  - par exemple : σ_3 définie par $t_1 \mapsto (t_2 \to t_2), t_3 \mapsto t_2$
  - par exemple : σ_3 définie par $t_1 \mapsto (t_2 \to t_2), t_3 \mapsto t_2, t_2 \mapsto t_8 \to t_8$
- un unifieur (ou solution) d'un système $(T_i, S_i)_{i \in I}$ est une substitution σ telle que pour chaque équation (T_i, S_i), $\sigma(T_i) = \sigma(S_i)$.
  - $\sigma(T)$ est le terme obtenu en remplaçant dans T chaque variable par son image par σ
  - σ_1 n'est pas un unifieur de S_1
  - σ_2 est un unifieur de S_1
  - σ_3 est un unifieur de S_2
  - σ_4 est un unifieur de S_2

Inférence de λ_{ST} : Unification (IV)

- un unifieur σ_0 (d'un système donné) est plus général qu'un unifieur σ_1 quand il existe σ_2 tel que $\sigma_1 = \sigma_0 \circ \sigma_2$
  - par exemple : σ_3 est plus général que σ_4
- un unifieur σ_0 (d'un système donné) est le plus général (mgu) s'il n'existe pas d'unifieur plus général que lui.
  - par exemple : σ_3 est un mgu de S_2
  - par exemple : σ_3 est l'unique mgu de S_1
- unifier un système d'équation, c'est trouver un mgu pour ce système, ou échouer en prouvant qu'il n'existe pas d'unificateur pour ce système.
  - par exemple : S_2 s'unifie en σ_3
  - par exemple : l'unification de $\{t_3 = t_2 \to t_3\}$ échoue.

Inférence de λ_{ST} : Unification (IV)

- Plusieurs algorithmes connus : W, Martelli-Montanari, Hindley-Milner, ...
- Principes communs :
  - on observe les équations une par une en maintenant un système (initialement, le système en entrée) et une substitution (initialement l'identité).
  - cas d'une équation (x, T) ou (T, x)
    - si $T = x$, on supprime l'équation du système,
    - si x est présent dans T et que $T \neq x$, on échoue.
    - sinon on compose $x \mapsto T$ avec σ (pour obtenir un nouveau σ), on supprime l'équation du système et on et applique le nouveau σ à toutes les équations du système.
  - cas d'une équation $(f(T_1, \ldots, T_n), g(T'_1, \ldots, T'_m))$
    - si $f \neq g$ on échoue.
    - sinon on ajoute au système les équations $\{(T_1, T'_1), \ldots (T_n, T'_m)\}$ (car $n = m$) et on supprime l'équation.

Inférence de λ_{ST} : Résolution de systèmes d'équation

(Méthode naïve, à raffiner en projet)

- On travaille sur un ensemble d'équations E.
- On identifie l'équation qui contient le but t_0
  - On conserve cette équation (qui donnera le type cherché)
- on prend une équation de l'ensemble :
  - Cas $t_i = T$
    - si T_i appartient strictement à T, alors on échoue (deux types non unifiables).
    - sinon on remplace T_i par T dans toutes les autres équations et on supprime celle-là.
  - Cas $T = t_i$
    - symétrique.
  - Cas $T \rightarrow T' = S \rightarrow S'$
    - on ajoute les équations $T = S$ et $T' = S'$
    - on supprime celle-là.
- Terminaison : (nombre de variables, nombre de flèches, nombre d'équations) décroit strictement pour l'ordre lexicographique.

Types Produits

- Un type produit est une produit cartésien de types.
- Idée : On ajoute un constructeur de couples à la syntaxe.

$$M ::= x \mid M\ M \mid \lambda x.M \mid (M, N)$$

- (M, N) permet à M et N de se réduire.

$$\textbf{(ProG)}\,\frac{M \longrightarrow M'}{(M, N) \longrightarrow (M', N')} \qquad \textbf{(ProD)}\,\frac{N \longrightarrow N'}{(M, N) \longrightarrow (M', N')}$$

- On type avec :

$$\textbf{(Pro)}\,\frac{\Gamma \vdash M : T \qquad \Gamma \vdash N : U}{\Gamma \vdash (M, N) : T \times U}$$

Types Produits

- Un type produit est une produit cartésien de types.
- Idée : On ajoute un constructeur de couples à la syntaxe.

$$M ::= x \mid M\ M \mid \lambda x.M \mid (M, N)$$

- (M, N) permet à M et N de se réduire.

$$(\textbf{ProG})\frac{M \longrightarrow M'}{(M, N) \longrightarrow (M', N')} \qquad (\textbf{ProD})\frac{N \longrightarrow N'}{(M, N) \longrightarrow (M', N')}$$

- On type avec :

$$(\textbf{Pro})\frac{\Gamma \vdash M : T \qquad \Gamma \vdash N : U}{\Gamma \vdash (M, N) : T \times U}$$

- Problèmes :
  - on ne peut pas déconstruire les couples

Types Produits (corrigés)

- On ajoute des déconstructeurs de couples à la syntaxe.

$$M ::= x \mid M\ M \mid \lambda x.M \mid (M, M) \mid \Pi_1\ M \mid \Pi_2\ M$$

- qui permettent de projeter un couple

$$(\textbf{ProG})\,\frac{M \longrightarrow M'}{(M, N) \longrightarrow (M', N')} \qquad (\textbf{ProD})\,\frac{N \longrightarrow N'}{(M, N) \longrightarrow (M', N')}$$

$$(\textbf{PjG})\,\frac{}{\Pi_1\ (M, N) \longrightarrow M} \qquad (\textbf{PjD})\,\frac{}{\Pi_2\ (M, N) \longrightarrow M}$$

- et que l'on doit typer

$$(\textbf{Pro})\,\frac{\Gamma \vdash M : T \qquad \Gamma \vdash N : U}{\Gamma \vdash (M, N) : T \times U} \qquad (\textbf{PjG})\,\frac{\Gamma \vdash M : T \times U}{\Gamma \vdash \Pi_1\ M : T}$$

$$(\textbf{PjD})\,\frac{\Gamma \vdash M : T \times U}{\Gamma \vdash \Pi_2\ M : U}$$

- Exemple :
  - Evaluer et typer $(\lambda c.(\Pi_1\ c)\ (\Pi_2\ c))\ (I, I\ K)$

Types Sommes

- Un type somme est une union de types.
- Idée : On ajoute une somme à la syntaxe.

$$M ::= x \mid M\ M \mid \lambda x.M \mid M + M$$

- $M + N$ se comporte de manière non-déterministe comme M ou N.

$$(\textbf{SumG})\frac{M \longrightarrow M'}{M + N \longrightarrow M'} \qquad (\textbf{SumD})\frac{N \longrightarrow N'}{M + N \longrightarrow N'}$$

- On type avec :

$$(\textbf{Sum})\frac{\Gamma \vdash M : T \qquad \Gamma \vdash N : U}{\Gamma \vdash M + N : T + U}$$

Types Sommes

- Un type somme est une union de types.
- Idée : On ajoute une somme à la syntaxe.

$$M ::= x \mid M\ M \mid \lambda x.M \mid M + M$$

- $M + N$ se comporte de manière non-déterministe comme M ou N.

$$(\textbf{SumG})\frac{M \longrightarrow M'}{M + N \longrightarrow M'} \qquad (\textbf{SumD})\frac{N \longrightarrow N'}{M + N \longrightarrow N'}$$

- On type avec :

$$(\textbf{Sum})\frac{\Gamma \vdash M : T \qquad \Gamma \vdash N : U}{\Gamma \vdash M + N : T + U}$$

- Problèmes :
  - la réduction asujettie est violée $I + K \longrightarrow I$.
  - on ne peut pas appliquer de fonction de manière intéressante : quel M peut apparaître dans $M\ (I + K)$
  - mauvaise idée.

Types Sommes (Corrigés)

- On a des constructeurs qui "indique un côté" et un destructeur qui branche.

$$M ::= x \mid M\ M \mid \lambda x.M \mid \mathtt{g} : M \mid \mathtt{d} : M \mid \mathtt{sw}\ M : M + M$$

- On branche en fonction du côté du terme.

$$(\mathtt{SumG})\frac{M \longrightarrow M'}{\mathtt{g} : M \longrightarrow \mathtt{g} : M'} \qquad (\mathtt{SumD})\frac{N \longrightarrow N'}{\mathtt{g} : N \longrightarrow \mathtt{g} : N'}$$

$$(\mathtt{SwG})\frac{}{\mathtt{sw}\ \mathtt{g} : M : N_1 + N_2 \longrightarrow N_1} \qquad (\mathtt{SwG})\frac{}{\mathtt{sw}\ \mathtt{d} : M : N_1 + N_2 \longrightarrow N_2}$$

- On type avec :

$$(\mathtt{SumG})\frac{\Gamma \vdash M : T}{\Gamma \vdash \mathtt{g} : M : T + U} \qquad (\mathtt{SumG})\frac{\Gamma \vdash M : U}{\Gamma \vdash \mathtt{d} : M : T + U}$$

$$(\mathtt{Sw})\frac{\Gamma \vdash M : T + U \qquad \Gamma \vdash N_1 : S \qquad \Gamma \vdash N_2 : S}{\Gamma \vdash \mathtt{sw}\ \mathtt{d} : M : N_1 + N_2 : S}$$

Types Sommes (Corrigés)

- On a des constructeurs qui "indique un côté" et un destructeur qui branche.

$$M ::= x \mid M\ M \mid \lambda x.M \mid \mathtt{g} : M \mid \mathtt{d} : M \mid \mathtt{sw}\ M : M + M$$

- On branche en fonction du côté du terme.

$$(\mathtt{SumG})\frac{M \longrightarrow M'}{\mathtt{g} : M \longrightarrow \mathtt{g} : M'} \qquad (\mathtt{SumD})\frac{N \longrightarrow N'}{\mathtt{g} : N \longrightarrow \mathtt{g} : N'}$$

$$(\mathtt{SwG})\frac{}{\mathtt{sw}\ \mathtt{g} : M : N_1 + N_2 \longrightarrow N_1} \qquad (\mathtt{SwG})\frac{}{\mathtt{sw}\ \mathtt{d} : M : N_1 + N_2 \longrightarrow N_2}$$

- On type avec :

$$(\mathtt{SumG})\frac{\Gamma \vdash M : T}{\Gamma \vdash \mathtt{g} : M : T + U} \qquad (\mathtt{SumG})\frac{\Gamma \vdash M : U}{\Gamma \vdash \mathtt{d} : M : T + U}$$

$$(\mathtt{Sw})\frac{\Gamma \vdash M : T + U \qquad \Gamma \vdash N_1 : S \qquad \Gamma \vdash N_2 : S}{\Gamma \vdash \mathtt{sw}\ \mathtt{d} : M : N_1 + N_2 : S}$$

- Problème :
  - On ne peut pas se servir de M dans chacune des deux branches.

Types Sommes (Corrigés II)

- Le destructeur lie une variable.

$$M ::= x \mid M\ M \mid \lambda x.M \mid \mathtt{g} : M \mid \mathtt{d} : M \mid \mathtt{sw}\ M\ \triangleright x : M + M$$

- On branche en fonction du côté du terme.

$$(\mathtt{SumG})\,\frac{M \longrightarrow M'}{\mathtt{g} : M \longrightarrow \mathtt{g} : M'} \qquad (\mathtt{SumD})\,\frac{N \longrightarrow N'}{\mathtt{g} : N \longrightarrow \mathtt{g} : N'}$$

$$(\mathtt{SwG})\,\frac{}{\mathtt{sw}\ \mathtt{g} : M\ \triangleright x : N_1 + N_2 \longrightarrow N_1[M/x]}$$

$$(\mathtt{SwG})\,\frac{}{\mathtt{sw}\ \mathtt{d} : M\ \triangleright x : N_1 + N_2 \longrightarrow N_2[M/x]}$$

- On type avec :

$$(\mathtt{SumG})\,\frac{\Gamma \vdash M : T}{\Gamma \vdash \mathtt{g} : M : T + U} \qquad (\mathtt{SumG})\,\frac{\Gamma \vdash M : U}{\Gamma \vdash \mathtt{d} : M : T + U}$$

$$(\mathtt{Sw})\,\frac{\Gamma \vdash M : T + U \qquad \Gamma, x : T \vdash N_1 : S \qquad \Gamma, x : U \vdash N_2 : S}{\Gamma \vdash \mathtt{sw}\ \mathtt{d} : M : N_1 + N_2 : S}$$

Types Sommes (Corrigés II)

- Le destructeur lie une variable.

$$M ::= x \mid M\ M \mid \lambda x.M \mid \mathtt{g} : M \mid \mathtt{d} : M \mid \mathtt{sw}\ M\ \triangleright x : M + M$$

- On branche en fonction du côté du terme.

$$(\mathtt{SumG})\ \frac{M \longrightarrow M'}{\mathtt{g} : M \longrightarrow \mathtt{g} : M'} \qquad (\mathtt{SumD})\ \frac{N \longrightarrow N'}{\mathtt{g} : N \longrightarrow \mathtt{g} : N'}$$

$$(\mathtt{SwG})\ \frac{}{\mathtt{sw}\ \mathtt{g} : M\ \triangleright x : N_1 + N_2 \longrightarrow N_1[M/x]}$$

$$(\mathtt{SwG})\ \frac{}{\mathtt{sw}\ \mathtt{d} : M\ \triangleright x : N_1 + N_2 \longrightarrow N_2[M/x]}$$

- On type avec :

$$(\mathtt{SumG})\ \frac{\Gamma \vdash M : T}{\Gamma \vdash \mathtt{g} : M : T + U} \qquad (\mathtt{SumG})\ \frac{\Gamma \vdash M : U}{\Gamma \vdash \mathtt{d} : M : T + U}$$

$$(\mathtt{Sw})\ \frac{\Gamma \vdash M : T + U \qquad \Gamma, x : T \vdash N_1 : S \qquad \Gamma, x : U \vdash N_2 : S}{\Gamma \vdash \mathtt{sw}\ \mathtt{d} : M : N_1 + N_2 : S}$$

- Exemple :
  - Evaluer et typer $(\lambda x.\mathtt{sw}\ x\ \triangleright y : y\ 2 + y\ 3\ 4)\ (\mathtt{g} : I)$.
  - Evaluer et typer $(\lambda x.\mathtt{sw}\ x\ \triangleright y : y\ 2 + y\ 3\ 4)\ (\mathtt{d} : K)$.

Polymorphisme

- Polymorphisme de généricité : une même fonction/méthode est utilisable avec des types différents.
- Dans λ_{ST}, un terme typable est typable avec une infinité de type :
  - $\emptyset \vdash I : \alpha \to \alpha$
  - $\emptyset \vdash I : (\alpha \to \alpha) \to (\alpha \to \alpha)$
  - $\emptyset \vdash I : (\alpha \to \beta \to \alpha) \to (\alpha \to \beta \to \alpha)$
  - $\emptyset \vdash I : (\alpha \to (\alpha \to \alpha) \to \alpha) \to (\alpha \to (\alpha \to \alpha) \to \alpha)$
  - ...
- Dans ce processus, chaque variable de type agit comme une variable mathématique, que l'on peut remplacer par n'importe quel type.
- Cette caractéristique permet de typer $I\ I$:
  -
$$\dfrac{\dfrac{\dfrac{}{x : \alpha \to \alpha \vdash x : \alpha \to \alpha}\,(\mathbf{Var})}{\emptyset \vdash I : (\alpha \to \alpha) \to (\alpha \to \alpha)}\,(\mathbf{Abs}) \qquad \dfrac{\dfrac{}{x : \alpha \vdash x : \alpha}\,(\mathbf{Var})}{\emptyset \vdash I : \alpha \to \alpha}\,(\mathbf{Abs})}{\emptyset \vdash I\ I : \alpha \to \alpha}\,(\mathbf{App})$$
  - Dans la branche de gauche, I est typée avec le type $(\alpha \to \alpha) \to (\alpha \to \alpha)$, et dans la branche de droite avec $\alpha \to \alpha$.

Polymorphisme (II)

- Ce processus ne s'applique pas aux variables.
- $\delta = \lambda x.x\ x$ n'est pas typable :

$$
\cfrac{\cfrac{\cfrac{}{x : t_1 \vdash x : t_3 \to T_2}\,(\textbf{Var}) \qquad \cfrac{}{x : t_1 \vdash x : t_3}\,(\textbf{Var})}{x : T_1 \vdash x\ x : T_2}\,(\textbf{App})}{\emptyset \vdash \lambda x.x\ x : t_1 \to t_2}\,(\textbf{Abs})
$$

- on tombe sur $t_3 \to t_2 = t_3$ qui n'est pas unifiable.
- la règle (**App**) force le contexte Γ (donc le type de x) à être le même des deux côtés de l'application.
- pourtant $\delta\ I$ ne semble pas plus problématique que $I\ I$.
- On peut étendre le système de types pour typer ces termes faisant apparaître de la généricité.
  - le même code qui va substituer les deux occurences de x est utilisé "de deux manières différentes".

Polymorphisme (III)

- On ajoute la quantification universelle aux types:

$$T ::= \alpha \mid T \to T \mid \forall \alpha . T$$

- on ajoute des règles pour ce constructeur de types :

$$(\mathbf{Gen})\frac{\Gamma \vdash M : T}{\Gamma \vdash M : \forall \alpha . T} \qquad (\mathbf{Inst})\frac{\Gamma \vdash M : \forall \alpha . T}{\Gamma \vdash M : T[U/\alpha]}$$

- (**Gen**) permet de généraliser les variables de types d'un type donné à un terme.
- (**Inst**) permet d'instantier la variable liée d'un type universel par n'importe quel type.
- c'est *Système F*.

Polymorphisme (IV)

- δ est typable :

$$\dfrac{\dfrac{\dfrac{}{x : \forall\alpha.\alpha \to \alpha \vdash x : (\forall\alpha.\alpha \to \alpha)}\,(\textbf{Var})}{x : \forall\alpha.\alpha \to \alpha \vdash x : (\forall\alpha.\alpha \to \alpha) \to (\forall\alpha.\alpha \to \alpha)}\,(\textbf{Inst}) \qquad \dfrac{}{x : \forall\alpha.\alpha \to \alpha \vdash x : \forall\alpha.\alpha \to \alpha}\,(\textbf{Var})}{\dfrac{x : \forall\alpha.\alpha \to \alpha \vdash x\ x : (\forall\alpha.\alpha \to \alpha)}{\emptyset \vdash \delta : (\forall\alpha.\alpha \to \alpha) \to (\forall\alpha.\alpha \to \alpha)}\,(\textbf{Abs})}\,(\textbf{App})$$

- ici on utilise la règle (**Inst**) pour remplacer chaque α par le type $\forall\alpha.\alpha \to \alpha$ lui-même.
- Ω n'est pas typable.
  - le type de δ ne peut pas être généralisé de manière intéressante (il n'a pas de variable libre)
  - on ne peut donc pas typer δ avec deux types différents dans $\delta\ \delta$.

Polymorphisme (V)

- Théorème : En Système F, M est typable si et seulement si M est fortement normalisant.
- Conséquence :

Polymorphisme (V)

- Théorème : En Système F, M est typable si et seulement si M est fortement normalisant.
- Conséquence : F n'est pas inférable (l'inférence c'est pas décidable)
  - elle n'est plus dirigée par la syntaxe, (on peut utiliser (**Abs**) ou (**Inst**))
  - quand on utilise (**Inst**), on a plusieurs possibilités pour U.
- pour un langages de programmation, on veut du polymorphisme inférable.

Let-polymorphisme

- les langages fonctionnels utilisent un polymorphisme plus restreint
- ils disposent d'une construction `let`
  - $\texttt{let}\ x = e_1\ \texttt{in}\ e_2$ a la même sémantique que $(\lambda x.e_2)\ e_1$
  - c'est le seul endroit où l'on peut créer du polymorphisme

$$(\texttt{Let})\frac{\Gamma \vdash e_1 : T \qquad \Gamma, e_1 : \mathbf{Gen}(T) \vdash e_2 : U}{\Gamma \vdash \texttt{let}\ x = e_1\ \texttt{in}\ e_2 : U}$$

    avec $\mathbf{Gen}(T)$ l'opération syntaxique qui transforme T en $\forall\alpha_1\forall\alpha_2\ldots\forall\alpha_n.T$ si $\alpha_1,\ldots,\alpha_n$ sont les variables de types libres de T.
  - l'inférence devient décidable.
- l'algorithme de *Hindley-Milner* (variante d'unification) est utilisé pour l'inférence de types des langages *à la ML*.

Typage impératif

- les langages ML contiennent des traits impératifs.
- On peut les ajouter à λ :
  - allocation `ref` M
  - déréférencement $!M$
  - réaffectation $M{:=}N$
- On ajoute un constructeur de type spécifique **Ref** pour les typer.
- Ca donne :

$$(\texttt{Ref})\frac{\Gamma \vdash M : T}{\Gamma \vdash \texttt{ref}\ T : \textbf{Ref}\ T} \qquad (\texttt{Deref})\frac{\Gamma \vdash M : \textbf{Ref}\ T}{\Gamma \vdash !M : T}$$

$$(\texttt{Assign})\frac{\Gamma \vdash M : \textbf{Ref}\ T \Gamma \vdash N : T}{\Gamma \vdash M{:=}N : ()}$$

Typage impératif (II)

- en fait ça ne fonctionne pas.

$$\begin{array}{l} \texttt{let}\ f = \texttt{ref}\ \lambda x.x\ \texttt{in} \\ \texttt{let}\ \_ = f := \lambda x.x + 1\ \texttt{in} \\ !f\ I \end{array}$$

- il ne faut pas toujours généraliser les types impératifs lors d'un `let`
- la solution des langages ML est :
  - distinguer les termes expansifs et non-expansifs
  - les termes non-expansifs sont généralisés,
  - les termes expansifs recoivent un polymorphisme faible
    - par exemple, **Ref** $\_\alpha \longrightarrow \_\alpha$
    - pendant l'inférence, la première fois qu'ils sont instantiés ils se transforment en leur instantiation : $\_\alpha \longrightarrow \_\alpha$ devient **Nat** $\longrightarrow$ **Nat** et ne peut plus être modifié.
    - en OCaml c'est `'a` vs. `'_a`
- Détail au Cours 04.

SORBONNE UNIVERSITÉ

Curry-Howard : Logique Intuitionniste

- Formules logiques avec l'implication.

$$\phi ::= A \mid \phi \Rightarrow \phi$$

- Γ : ensemble d'hypothèses (de formules),
- Jugements $\Gamma \vdash \phi$: "ϕ est prouvable avec les hypothèses Γ"
- Séquents Intuitionnistes :

$$(\mathbf{Ax})\frac{}{\Gamma, A \vdash A} \qquad (\mathbf{MP})\frac{\Gamma \vdash A \Rightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B} \qquad (\mathbf{I})\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}$$

- Formules prouvables :

$$(\mathbf{I})\dfrac{(\mathbf{I})\dfrac{(\mathbf{I})\dfrac{(\mathbf{MP})\dfrac{(\mathbf{MP})\dfrac{(\mathbf{Ax})\dfrac{}{\Gamma \vdash A \Rightarrow (B \Rightarrow C)} \quad (\mathbf{Ax})\dfrac{}{\Gamma \vdash A}}{\Gamma \vdash B \Rightarrow C} \qquad (\mathbf{MP})\dfrac{(\mathbf{Ax})\dfrac{}{\Gamma \vdash A \Rightarrow B} \quad (\mathbf{Ax})\dfrac{}{\Gamma \vdash A}}{\Gamma \vdash B}}{A \Rightarrow B \Rightarrow C, A \Rightarrow B, A \vdash C}}{A \Rightarrow B \Rightarrow C, A \Rightarrow B \vdash A \Rightarrow C}}{A \Rightarrow B \Rightarrow C \vdash (A \Rightarrow B) \Rightarrow A \Rightarrow C}}{\vdash (A \Rightarrow B \Rightarrow C) \Rightarrow (A \Rightarrow B) \Rightarrow A \Rightarrow C}$$

Curry-Howard : Correspondance

- Formules de SI $\leftrightarrow$ Types de λ_{ST}
- Preuves de SI $\leftrightarrow$ Dérivation de typage de λ_{ST}
  - $=$ Termes de λ_{ST}
  - car le système de types est dirigé par la syntaxe

$$
(\textbf{Abs})\,\dfrac{(\textbf{Abs})\,\dfrac{(\textbf{Abs})\,\dfrac{(\textbf{App})\,\dfrac{(\textbf{App})\,\dfrac{(\textbf{Var})\,\dfrac{}{\Gamma \vdash x : A \to (B \to C)} \quad (\textbf{Var})\,\dfrac{}{\Gamma \vdash z : A}}{\Gamma \vdash x\ z : B \to C} \quad (\textbf{App})\,\dfrac{(\textbf{Var})\,\dfrac{}{\Gamma \vdash y : A \to B} \quad (\textbf{Var})\,\dfrac{}{\Gamma \vdash z : A}}{\Gamma \vdash y\ z : B}}{x : A \to B \to C, y : A \to B, z : A \vdash \lambda(x\ z)\ (y\ z) : C}}{x : A \to B \to C, y : A \to B \vdash \lambda z.(x\ z)\ (y\ z) : A \to C}}{x : A \to B \to C \vdash \lambda yz.(x\ z)\ (y\ z) : (A \to B) \to A \to C}}{\vdash S : (A \to B \to C) \to (A \to B) \to A \Rightarrow C}
$$

- ??? de SI $\leftrightarrow$ Réduction de λ_{ST}

Curry-Howard : Elimination des coupures

- Opération de transformation des preuves.
- Utilisation d'un lemme :

$$
(\textbf{App})\,\dfrac{(\textbf{I})\,\dfrac{\dfrac{\mathcal{P}_1}{\Gamma, A \vdash B}}{\Gamma \vdash A \Rightarrow B} \qquad \dfrac{\mathcal{P}_2}{\Gamma \vdash A}}{\Gamma \vdash B} \qquad \longrightarrow \qquad \dfrac{\mathcal{P}_1[\mathcal{P}_2]}{\Gamma \vdash B}
$$

avec $\mathcal{P}_1[\mathcal{P}_2]$ la preuve obtenue en prenant $\mathcal{P}_1$ et en remplaçant tous les $(\textbf{Ax})\dfrac{}{\Gamma', A \vdash A}$ par $\dfrac{\mathcal{P}_2}{\Gamma' \vdash A}$

- on simplifie une preuve en *inlinant* un lemme à tous les endroits où on en avait besoin.
- l'enchainement de (**App**) avec (**I**) à gauche s'appelle une coupure, le processus correspondant à la réduction est l'élimination des coupures.
- éliminer une coupure peut faire grossir la preuve et ajouter des coupures.
  - en copiant plusieurs fois les coupures dans $\mathcal{P}_\in$
- l'élimination des coupures termine.

Curry-Howard : Logique Classique

- Calcul des Séquents classique :

$$(\textbf{TE})\frac{}{\vdash A \vee A} \qquad \textbf{ou} \qquad (\textbf{Abs})\frac{\Gamma \vdash \neg\neg A}{\Gamma \vdash A} \qquad \textbf{ou}$$

$$(\textbf{PL})\frac{\Gamma \vdash ((A \Rightarrow B) \Rightarrow A) \Rightarrow A}{\Gamma \vdash A}$$

- comment introduire $A \vee B$ dans λ_{ST} ?

Curry-Howard : Logique Classique

- Calcul des Séquents classique :

$$(\textbf{TE})\frac{}{\vdash A \vee A} \qquad \textbf{ou} \qquad (\textbf{Abs})\frac{\Gamma \vdash \neg\neg A}{\Gamma \vdash A} \qquad \textbf{ou}$$

$$(\textbf{PL})\frac{\Gamma \vdash ((A \Rightarrow B) \Rightarrow A) \Rightarrow A}{\Gamma \vdash A}$$

- comment introduire $A \vee B$ dans λ_{ST} ?

$$(\texttt{SumG})\frac{\Gamma \vdash M : T}{\Gamma \vdash \texttt{g} : M : T \vee U} \qquad (\texttt{SumG})\frac{\Gamma \vdash M : U}{\Gamma \vdash \texttt{d} : M : T \vee U}$$

$$(\texttt{Sw})\frac{\Gamma \vdash M : T \vee U \qquad \Gamma, x : T \vdash N_1 : S \qquad \Gamma, x : U \vdash N_2 : S}{\Gamma \vdash \texttt{sw d} : M : N_1 + N_2 : S}$$

soit

$$(\textbf{LI})\frac{\Gamma \vdash T}{\Gamma \vdash T \vee U} \qquad (\textbf{RI})\frac{\Gamma \vdash U}{\Gamma \vdash T \vee U} \qquad (\textbf{E})\frac{\Gamma \vdash T \vee U \qquad \Gamma, T \vdash S \qquad \Gamma, U \vdash S}{\Gamma \vdash S}$$

Curry-Howard : Logique Classique (II)

- il faut complexifier λ pour obtenir des calculs en relation avec SK
- le $\lambda\mu$-calcul permet de définir des termes nommés:

$$M ::= x \mid \lambda x.M \mid M\ M \mid \mu\alpha.E \qquad\qquad E ::= [\alpha]\ M$$

- la réduction structurelle permet, dans un terme liant le nom α, de distribuer un argument N à tous les sous-termes nommés par α.

$$\frac{}{(\mu\alpha.M)\ N \longrightarrow \mu\alpha.M[[\alpha](N\ M')/[\alpha]M']}$$

- le système de types standard de $\lambda\mu$ correspond à la déduction naturelle classique.
- $\overline{\lambda}\mu\tilde{\mu}$ correspond aux séquents classiques

$$C ::= [V|E] \qquad V ::= x \mid \lambda x.V \mid e\ v \mid \mu\alpha.c \qquad E ::= \alpha \mid \alpha\lambda.e \mid v\ e \mid \overline{\mu}x.c$$

- avec la réduction

$$\frac{}{[\lambda x.V|V'\ E] \longrightarrow [V'|\overline{\mu}x.[V|E]]} \qquad \frac{}{[E'\ V|\alpha\lambda.E] \longrightarrow [\mu\alpha.[V|E]|E'}$$

$$\overline{\mu\alpha.[V|\alpha]} \longrightarrow V \qquad \frac{}{[\mu\alpha.C|E] \longrightarrow C[E/\alpha]} \qquad \frac{}{[V|\overline{\mu}.C] \longrightarrow C[V/x]}$$

$$\overline{\overline{\mu}x.[x|E]} \longrightarrow E$$

- termes, contextes, et commandes manipulent le flot de contrôle.

Cube de Barendregt

- trois directions pour enrichir λ_{ST}
- termes dépendants de types : Polymorphisme
  - Système F présenté avec instantiation explicite et réduction typée

$$M ::= x \mid \lambda x.M \mid M\ M \mid \Lambda T.M \mid T \qquad T ::= \alpha \mid T \rightarrow T \mid \forall\alpha.T$$

$$\frac{\Gamma \vdash \Lambda X.M : \forall\alpha.T}{\Gamma \vdash \Lambda X.M\ U \longrightarrow \Gamma \vdash M : T[U/X]}$$

  - on peut définir δ comme $\Lambda Y.\lambda x.(x\ Y \rightarrow Y)\ (x\ Y)$
  - et I comme $\Lambda X.\lambda x.x$
  - et typer $\delta\ (\forall\alpha.(\alpha \rightarrow \alpha))\ I$.
- types dépendant de termes (appelés "types dépendants")
  - formellement $T ::= \alpha \mid T \rightarrow T \mid \pi x.T$
  - permet de définir, par exemple, 4 `vect`, les vecteurs de taille inférieure à 4.
- types dépendant de types (constructeurs de types)
  - formellement $T ::= \alpha \mid T \rightarrow T \mid \Pi X.T$
  - permet de définir `Liste A`, les listes d'éléments de types A,
  - hierarchie de sortes (comme en PAF)

Curry-Howard : Correspondances

- Le Calcul des Constructions *ferme* le cube de Barendregt
- Quelques correspondances de Curry-Howard connues :
  - SI $\leftrightarrow$ λ_{ST}
  - SK $\leftrightarrow$ $\overline{\lambda}\mu\tilde{\mu}$
  - Arithmétique de Peano $\leftrightarrow$ Système F
  - Logique de Hilbert $\leftrightarrow$ Logique Combinatoire
  - LI d'ordre supérieur $\leftrightarrow$ Calcul des Constructions.
  - λ linéaires $\leftrightarrow$ Logiques Linéaires.

Recherche : Métiers et Structures en France

- Métiers :
  - Enseignants-Chercheurs (192 heqtd par an), deux corps : Maîtres de Conférence et Professeur
  - Chercheurs rattachés à un institut (CNRS, INRIA)
  - Ingénieurs.
  - Non-fonctionnaires (doctorants, post-doc, ATER, CDD, ...)
- Recrutement :
  - EC: Doctorat + Qualification + Concours locaux
  - C: Doctorat + Concours nationaux
  - I: Concours nationaux
- Hierarchie des structures :
  - équipe : 10, même thématique.
  - laboratoire : 100, gestion financière (hors RH).
  - Unité de Formation et de Recherche: 100, gestion financière RH,
  - Université: 1000, indépendance administrative et financière.
- Ailleurs : plus ou moins similaire (*tenure*, absence de chercheurs, contrats cours, ...)

Recherche : Financement et Evaluation

- Financement:
  - RH :
    - fonctionnaires : grille de salaires publique, ancienneté, point d'indice.
    - non-fonctionnaires : variables.
  - Autres (matériel, contrats courts - doctorats, missions) :
    - fonds propres laboratoire.
    - projets (région, France, Europe) : appels à candidatures.
- Evaluation:
  - évaluation des laboratoires, équipes, université tous les 5-7 ans,
  - HCERES : instance indépendante

Recherche : Articles

Chronologie :

- Ecriture d'un projet de financement sur un sujet donné.
- Travail en collaboration (inter ou intra université) sur le sujet.
- Découverte de résultats.
- Rédaction d'un article (10-15 pages) rassemblant résultats et preuves..
- Soumission de l'article au comité d'une conférence.
- Acceptation de l'article, présentation à la conférence, publication des *proceedings*.
- Rédaction d'une version longue (40 pages).
- Soumission de la version longue au comité d'un journal.
- Acceptation de l'article et publication du journal.

Recherche : Articles (II)

- Publication :
  - surtout numérique,
  - problèmes de droits (disponibilité des articles).
  - articles (*preprint*) (parfois vidéos, transparents) disponibles sur les sites personnels des auteurs.
- Acceptation :
  - conférences et journaux ont un *Program Committee* (des chercheurs du domaine) qui
    - distribue les articles à des *reviewers* (des chercheurs du sous-domaine),
    - collecte les avis,
    - décide de l'acceptation.
  - principe de la recherche *peer-reviewed*
  - limites de la recherche moderne (*réfutabilité* Popperienne)
- des sites (DBLP, par exemple) répertorient les publications (acceptées) des chercheurs.

Article : Composition

- *Abstract* : un résumé de l'article (objectifs, méthodes, résultats)
- Introduction (Section 1) : explique en détail de manière informelle le contexte, les objectifs, les méthodes de l'article.
- Formalisme (Section 2) : introduit le langage (mathématique ou informatique) étudié
- Coeur de l'article (Section 3+) : présente les techniques utilisée en détail et les résultats (théorèmes, algorithmes).
- Preuves (version longue) : souvent absentes dans l'article de conférence.
- *Related Works* : mise en relation de l'article avec les autres travaux du domaine (précédents, ou concurrents).
- Bibliographie : détails de tous les articles cités.

Synthèse d'article : Lecture

- Auteurs : comprendre leurs domaines et leurs travaux précédents
- *Abstract* : lire et comprendre pour choisir l'article.
- Introduction (Section 1) : comprendre les enjeux et les contributions.
- Formalisme (Section 2) : assimiler le formalisme (syntaxe, règles, exemples), produire des exemples
- Coeur de l'article (Section 3+) : comprendre les grandes lignes des méthodes utilisées, étudier des exemples, détailler les parties techniques.
- Preuves (version longue) : observer les techniques de preuve utilisées (ne pas lire les détails).
- *Related Works* : lire (au moins) l'abstract des articles cités, comprendre l'originalité de l'article.

Synthèse d'article : Consignes

- travail en binômes (note pouvant être différenciée).
- 25% de la note d'UE (moitié de la partie Typage),
- lire un article scientifique récent et en faire une synthèse,
- synthèse présentée sous formes :
  - d'un rapport (8-15 pages)
  - d'une soutenance (8 minutes devant les autres étudiants).
- choix d'un article dans une liste par Moodle:
  - articles de l'année 2024,
  - conférences internationales de programmation,
  - plus ou moins liés au typage.
  - les articles sont inéquitables en difficulté (pris en comptes dans la notation),

Synthèse d'article : Consignes

- Objectif : montrer que l'article est principalement compris.
- Ne pas traduire l'article.
- Ne pas sélectionner un sous-ensemble de l'article.
- Ne pas reprendre de phrases telles quelles.
- Reformuler les notions.
- Réécrire formules et exemples.
- Utiliser des exemples personnels inédits.
- Mettre en relation avec les connaissances acquises (cours de TAS, autres cours)
- Explorer les implémentations.
- Eventuellement, contacter les auteurs.

Synthèse d'article : Consignes (II)

- Rapport :
  - attention au plagiat (auteurs du rapport $\neq$ auteurs de l'article)
  - suivre la structure de l'article (mais ne pas recopier le contenu)
  - détailler ce qui a été compris (reformulation, exemples)
  - résumer ce qui n'a pas été compris.
  - critiquer (vis-à-vis du cours, du monde professionnel, ...) les contributions de l'article.
- Soutenance :
  - présenter les grandes lignes de l'article et le formalisme,
  - montrer quelques points techniques
  - s'adresser au public de TAS (cours connu)
  - soigner la présentation orale.

Conclusion

- TDs 03-06 - Travail en autonomie
  - réalisation d'un évaluateur-typeur,
  - synthèse d'article.
- Cours 04 : Typage en ML.