Projet fonctionnel

Basé sur le cours de Daniel Hirschkoff
Notes prises par Hugo Salou

ENS DE LYON

28 mars 2025

Table des matières

Introduction.

Le coups se décompose en deux parties :

- une partie *pratique* ($\sim 2/3$ du cours) avec de la programmation OCaml en binôme (fouine, puis pieuvre) ;
- une partie *théorique* : avec du λ-calcul, typage et lien avec la logique.

Ce document contiendra les notes de cours pour la partie théorique. D'autres documents pour la partie projet sont en ligne sur mon site (transformation CPS, et optimisations associées à la β-réduction).

Parfois, des références aux chapitres du cours de théorie de la programmation seront fait, ce sont des liens cliquables qui mènent vers le PDF du chapitre en question, par exemple :

Théorie de la Programmation [Chapitre 0].

1 Le λ-calcul pur.

Le λ-calcul a trois domaines proches :

- ▷ la *calculabilité*, avec l'équivalence entre machines de Turing et λ-expression (vue en FDI) ;
- ▷ la *programmation fonctionnelle* (vue en **Théorie de la Programmation** [Chapitre 6] avec le petit langage FUN) ;
- ▷ la *théorie de la démonstration* (vue dans la suite de ce cours).

On se donne un ensemble infini $\mathcal{V}$ de variables notées $x, y, z, \ldots$ Les *termes* (du λ-calcul) ou λ-termes sont définis par la grammaire

$$M, N, \ldots ::= \lambda x.\, M \mid M\ N \mid x.$$

La construction $\lambda x.\, M$ s'appelle l' *abstraction* ou λ-*abstraction*. Elle était notée $\texttt{fun}\, x \to M$ en cours de théorie de la programmation.

Notation.
- ▷ On notera $M\ N\ P$ pour $(M\ N)\ P$.
- ▷ On notera $\lambda xyz.M$ pour $\lambda x.\lambda y.\lambda z M$ (il n'y a pas lieu de mettre des parenthèses ici, vu qu'il n'y a pas d'ambigüités).
- ▷ On notera $\lambda x.\, M\ N$ pour $\lambda x.\, (M\ N)$. ***Attention***, c'est différent de $(\lambda x.\, M)\ N$.

1.1 Liaison et α-conversion.

Remarque 1.1 (Liaison). Le « λ » est un lieur. Dans $\lambda y.\, x\ y$, la variable y est *liée* mais pas x (la variable x libre). On note $\mathcal{VL}(M)$ l'ensemble des variables libres de M, définie par induction sur M (il y a *3* cas).

Remarque 1.2 (α-conversion).
On note $=_\alpha$ la relation d' *α-conversion*. C'est une relation binaire sur les λ-termes fondée sur l'idée de renommage des abstractions ***en évitant la capture de variables libres*** :

$$\lambda x.\, x\, y =_\alpha \lambda t.\, x\, t \neq_\alpha \lambda x.x\, x.$$

Ainsi $\lambda x.\, M =_\alpha \lambda z.\, M'$ où M' est obtenu en remplaçant x par z ***là où il apparaît libre*** et ***à condition que*** $z \notin \mathcal{V}\ell(M)$. Ceci, on peut le faire partout.

Lemme 1.1. La relation $=_\alpha$ est une relation d'équivalence. Si $M =_\alpha N$ alors $\mathcal{V}\ell(M) = \mathcal{V}\ell(N)$.

Par convention, on peut identifier les termes modulo $=_\alpha$. On pourra donc toujours dire

« considérons $\lambda x.\, M$ où $x \notin E$ [...] »

avec E un ensemble ***fini*** de variables.

Ceci veut dire qu'on notera

$$M = N \text{ pour signifier que } M =_\alpha N.$$

1.2 La β-réduction.

Définition 1.1 (β-réduction). On définit la relation de β-réduction sur les λ-termes, notée $\to_\beta$ ou $\to$, définie par les règles d'inférences :

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

$$\frac{M \to_\beta M'}{M\, N \to_\beta M'\, N} \qquad \frac{N \to_\beta N'}{M\, N \to_\beta M\, N'}$$

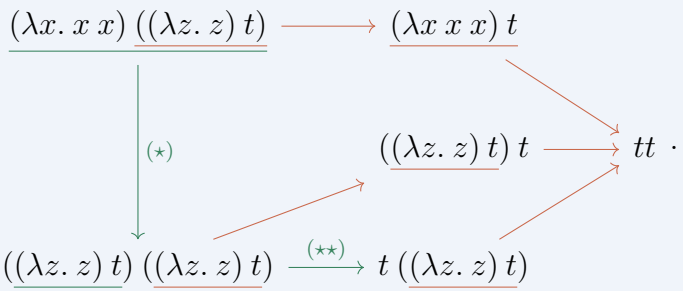
où $M[^N/_x]$ est la substitution de x par N dans M (on le défini ci-après).

Définition 1.2. Un terme de la forme $(\lambda x.\, M)\, N$ est appelé un *redex* (pour *reducible expression*) ou *β-redex*. Un terme M est une *forme normale* s'il n'existe pas de N tel que $M \to_\beta N$.

Remarque 1.3. La relation $\to_\beta$ n'est pas terminante :

$$\Omega := (\lambda x.\, x\, x)\, (\lambda y.\, y\, y) \to_\beta (\lambda y.\, y\, y)\, (\lambda y.\, y\, y) =_\alpha \Omega.$$

Exemple 1.1.



Un pas de β-réduction peut :

- ▷ dupliquer un terme (*c.f.* $(\star)$) ;
- ▷ laisser un redex inchangé (*c.f.* $(\star\star)$) ;
- ▷ faire disparaître un redex (qui n'est pas celui que l'on contracte) :

$$(\lambda x.\, u)((\lambda z.\, z)\, t) \ \to_\beta u\ ;$$

- ▷ créer de nouveaux redex :

$$(\lambda x.\, x\, y)\, (\lambda z.\, z) \to_\beta (\lambda z.\, z)\, y.$$

1.3 Substitutions.

Exemple 1.2. Le terme $\lambda xy.x$ c'est une « fonction fabriquant des fonctions constantes » au sens où

$$(\lambda xy.\ x)M \to_\beta \lambda y.\ M,$$

à condition que $y \notin \mathcal{VL}(M)$. On doit cependant α-renommer pour éviter la capture :

$$\begin{array}{c}(\lambda xy.x)\ (\lambda t.\ y) \not\to_\beta \lambda y.\ (\lambda t.\ y)\\ \qquad\quad \wr\| \\ (\lambda xy'.x)\ (\lambda t.\ y) \to_\beta \lambda y'.\ (\lambda t.\ y).\end{array}$$

Définition 1.3. On procède par induction, il y a trois cas :

- $y[N/x] := \begin{cases} N & \text{si } y = x \\ y & \text{si } y \neq x \end{cases}$
- $(M_1\ M_2)[N/x] := (M_1[N/x])\ (M_2[N/x])$
- $(\lambda y.\ M)[N/x] := \lambda y.\ (M[N/x])$ ***à condition que*** $y \notin \mathcal{VL}(N)$ ***et*** $y \neq x$.

Lemme 1.2 (Gymnastique des substitutions)**.** Pour $y \notin \mathcal{VL}(R)$,

$$(P[Q/y])[R/x] = (P[R/x])[Q[R/x]/y].$$

Lemme 1.3. Si $M \to_\beta M'$ alors $\mathcal{VL}(M') \subseteq \mathcal{VL}(M)$.

1.4 Comparaison λ-calcul et FUN.

En λ-calcul, on a une règle

$$\frac{M \to_\beta M'}{\lambda x.\ M \to_\beta \lambda x.\ M'.}$$

Cette règle n'existe pas en FUN (ni en fouine) car on traite les fonctions comme des valeurs. Et, en FUN, les trois règles suivantes sont

mutuellement exclusives :

$$\frac{}{(\lambda x.\, M)\; N \to_\beta M[^N/_x]} \qquad \frac{M \to_\beta M'}{M\; N \to_\beta M'\; N} \qquad \frac{N \to_\beta N'}{M\; N \to_\beta M\; N'}$$

car on attend que N soit une ***valeur*** avant de substituer.

En FUN (comme en fouine), pour l'exemple 1.1, on se limite à n'utiliser que les flèches rouges.

La relation $\to_\beta$ est donc « plus riche » que $\to_{\mathsf{FUN}}$. En FUN, on a une *stratégie de réduction* : on a au plus un redex qui peut être contracté. On n'a pas de notion de valeur en λ-calcul pur. Le « *résultat d'un calcul* » est une forme normale.

1.5 Exercice : les booléens.

On définit

$$\mathbf{T} := \lambda xy.\, x \qquad \mathbf{F} := \lambda xy.\, y.$$

Ainsi, pour tout M (si $y \notin \mathcal{V}\ell(M)$),

$$\mathbf{T}\; M \to \lambda y.\, M \qquad \mathbf{F}\; M \to \lambda y.\, y =: \mathbf{I}.$$

La construction `if` b `then` M `else` N se traduit en $b\; M\; N$.

Le « non » booléen peut se définir par :

- ▷ $\mathbf{not} := \lambda b.\, b\; \mathbf{F}\; \mathbf{T} = \lambda b.\, b\; (\lambda xy.\, y)\; (\lambda tu.\, t)$;
- ▷ $\mathbf{not}' := \lambda b.\, \lambda xy.\, byx$.

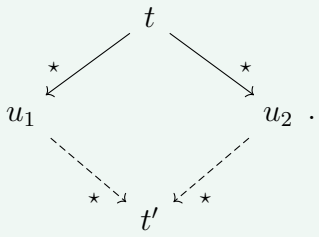
La première version est plus abstraite, la seconde est « plus électricien ». On a deux formes normales ***différentes***.

De même, on peut définir le « et » booléen :

- ▷ $\mathbf{and} := \lambda b_1.\, \lambda b_2.\, b_1\; (b_2\; \mathbf{T}\; \mathbf{F})\; \mathbf{F}$;
- ▷ $\mathbf{and}' := \lambda b_1.\, \lambda b_2.\, \lambda xy.\, b_1\; (b_2\; x\; y)\; y$.

1.6 Confluence de la β-réduction.

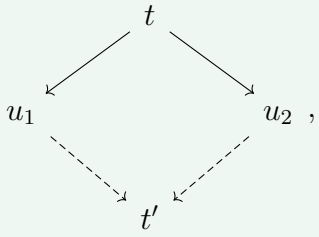
Définition 1.4 (Rappel, *c.f.* **Théorie de la Programmation** [Chapitre 10])**.** On dit que $\to$ est *confluente* en $t \in A$ si, dès que $t \to^\star u_1$ et $t \to^\star u_2$ il existe t' tel que $u_1 \to^\star t'$ et $u_2 \to^\star t'$.



Les flèches en pointillés représentent l'existence.

On dit que $\to$ est *confluente* si $\to$ est confluente en tout $a \in A$.

La propriété du diamant correspond au diagramme ci-dessous :



c'est-à-dire si $t \to u_1$ et $t \to u_2$ alors il existe t' tel que $u_1 \to t'$ et $u_2 \to t'$.

La confluence pour $\to$, c'est la propriété du diamant pour $\to^\star$. On sait déjà que la β-réduction n'a pas la propriété du diamant (certains chemins de l'exécution sont plus longs), mais on va montrer qu'elle est confluente.

Définition 1.5. On définit la relation de *réduction parallèle*, notée $\rightrightarrows$, par les règles d'inférences suivantes :

$$\frac{}{x \rightrightarrows x} \qquad \frac{M \rightrightarrows M'}{\lambda x.\, M \rightrightarrows \lambda x.\, M'}$$

$$\frac{M \rightrightarrows M' \quad N \rightrightarrows N'}{M\ N \rightrightarrows M'\ N'} \qquad \frac{M \rightrightarrows M' \quad N \rightrightarrows N'}{(\lambda x.\, M)\ N \rightrightarrows M'[N'/x]}$$

Lemme 1.4. La relation $\rightrightarrows$ est réflexive.

Lemme 1.5. Si $\mathscr{R} \subseteq \mathscr{S}$ alors $\mathscr{R}^\star \subseteq \mathscr{S}^\star$. De plus, $(\mathscr{R}^\star)^\star = \mathscr{R}^\star$.

Lemme 1.6. Les relations $\to^\star$ et $\rightrightarrows^\star$ coïncident.

Preuve.
- ▷ On a $\to^\star \subseteq \rightrightarrows^\star$ car cela découle de $\to \subseteq \rightrightarrows$ par induction sur $\to$ en utilisant la réflexivité de $\rightrightarrows$.
- ▷ On a $\rightrightarrows^\star \subseteq \to^\star$ car cela découle de $\rightrightarrows \subseteq \to^\star$. En effet, on montre que pour tout M, M' si $M \rightrightarrows M'$ alors $M \to^\star M'$, par induction sur $\rightrightarrows$. Il y a 4 cas.
  - Pour $x \rightrightarrows x$, c'est immédiat.
  - Pour l'abstraction, on suppose $M \rightrightarrows M'$ alors par induction $M \to^\star M'$, et donc $\lambda x.\, M \to^\star \lambda x.\, M'$ par induction sur $M \to^\star M'$.
  - Pour l'application, c'est plus simple que pour la précédente.
  - Pour la substitution, supposons $M \rightrightarrows M'$ et $N \rightrightarrows N$. On déduit par hypothèse d'induction $M \to^\star M'$ et $N \to^\star N'$. Et, par induction sur $M \to^\star M'$, on peut montrer que $(\lambda x.\, M)\ NN \to^\star (\lambda x.\, M')\ N$. Puis, par induction sur $N \to^\star N'$, on montre $(\lambda x.\, M')\ N \to^\star (\lambda x.\, M')\ N'$. Enfin, par la règle de β-réduction, on a $(\lambda x.\, M')N' \to M'[N'/x]$. On rassemble tout pour

obtenir :

$$(\lambda x.\, M)\, N \to^\star M'[N'/x].$$

□

On est donc ramené à montrer que $\rightrightarrows^\star$ a la propriété du diamant. Or $\rightrightarrows$ a la propriété du diamant, ce que l'on va montrer en TD.

Lemme 1.7. Si $M \rightrightarrows M'$ alors $N \rightrightarrows N'$ implique $M[N/x] \rightrightarrows M'[N'/x]$.

Preuve. Par induction sur $M \rightrightarrows M'$, il y a 4 cas. On ne traite que le cas de la 4ème règle. On suppose donc $M = (\lambda y.\, P)\, Q$ avec $y \notin \mathcal{V}\ell(N)$ et $y \neq x$. On suppose aussi $P \rightrightarrows P'$, $Q \rightrightarrows Q'$ et $M' = P'[Q'/y]$. On suppose de plus $N \rightrightarrows N'$. Par hypothèse d'induction, on a $P[N/x] \rightrightarrows P'[N'/x]$ et $Q[N/x] \rightrightarrows Q'[N'/x]$. On applique la 4ème règle d'inférence définissant $\rightrightarrows$ pour déduire

$$\underbrace{(\lambda y.\, (P[N/x]))}_{\substack{\| \\ (\lambda y.\, P)[N/x] \\ \text{car } x \neq y}}(Q[N/x]) \rightrightarrows (P'[N'/x])[Q'[N'/x]/y] = (P'[Q'/y])[N'/x]$$

par le lemme de gymnastique des substitutions et car $y \notin \mathcal{V}\ell(N') \subseteq \mathcal{V}\ell(N)$ et car $N \to^\star N'$. □

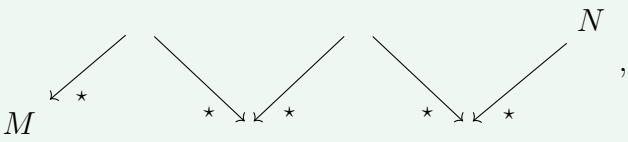
Proposition 1.1. La relation $\rightrightarrows$ a la propriété du diamant.

Preuve. Vu en TD. □

Corollaire 1.1. On a la confluence de $\to_\beta$.

Définition 1.6. La *β-équivalence*, ou *β-convertibilité* est la plus petite relation d'équivalence contenant $\to_\beta$. On la note $=_\beta$.

Si l'on a



alors $M =_\beta N$.

Proposition 1.2. Tout λ-terme est β-équivalent à au plus une forme normale.

Preuve. Si $M =_\beta N$ et M, N sont des formes normales, alors par confluence il existe P tel que $M \to^\star P$ et $N \to^\star P$. On a donc que $M = N = P$. □

Remarque 1.4 (Conséquences).
- ▷ Deux normales distinctes (au sens de $=_\alpha$) ne sont pas β-convertibles.
- ▷ Si on a un λ-terme qui diverge et qui a une forme normale, par exemple $(\lambda x.\ y)\ \Omega$, alors on peut toujours « revenir » sur la forme normale.

2 Le λ-calcul simplement typé.

Dans ce chapitre, on va parler de *typage*. Ceci permet de « stratifier » les λ-termes. En effet, pour l'instant, tous les termes se ressemblent.

2.1 Définition du système de types.

Définition 2.1. On se donne un ensemble de *types de base*, notés X, Y, Z, ... Les types simples sont donnés par la grammaire suivante :

$$A, B, C ::= \boldsymbol{X} \mid A \to B.$$

Il n'y a donc que deux « types » de types : les types de base, et les types fonctions. Il n'y a donc pas de type `unit`, `bool`, ... En effet, ceci demanderai d'ajouter des constantes `()`, `true`, `false`, *etc* dans la grammaire du λ-calcul (et ceci demanderai ensuite d'ajouter des règles de typage supplémentaire). On verra en TD comment typer **T** et **F** comme défini au chapitre précédent.

Par convention, on notera $A \to B \to C$ pour $A \to (B \to C)$.

Définition 2.2. On définit une *hypothèse de typage* comme un couple variable-type (x, A) noté $x : A$.

Définition 2.3. Un *environnent de typage*, noté Γ, Γ', *etc* est un dictionnaire sur $(\mathscr{V}, \mathsf{Types})$, *c.f.* cours de Théorie de la Programmation. On notera $\Gamma(x) = A$ lorsque Γ associe x à A. On définit

le *domaine* de Γ comme

$$\mathrm{dom}(\Gamma) := \{x \mid \exists A,\ \Gamma(x) = A\}.$$

On note aussi $\Gamma, x : A$ l'extension de Γ avec $x : A$ si $x \notin \mathrm{dom}(\Gamma)$.

Définition 2.4. On définit la *relation de typage*, notée $\Gamma \vdash M : A$ (« sous les hypothèses Γ, le λ-terme M a le type A ») par les règles d'inférences suivantes :

$$\Gamma(x) = A \ \frac{}{\Gamma \vdash x : A} \qquad \frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B}$$

$$x \notin \mathrm{dom}(\Gamma)\ \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.\ M : A \to B}$$

Dans cette dernière règle, on peut toujours l'appliquer modulo α-conversion (il suffit d'α-renommer x dans $\lambda x.\ M$).

Exemple 2.1. On peut omettre le « $\emptyset$ » avant « $\vdash$ ».

$$\dfrac{\dfrac{\dfrac{\dfrac{}{f : \boldsymbol{X} \to \boldsymbol{X}, z : \boldsymbol{X} \vdash f : \boldsymbol{X} \to \boldsymbol{X}} \quad \dfrac{\dfrac{}{f : \boldsymbol{X} \to \boldsymbol{X}, z : \boldsymbol{X} \vdash f : \boldsymbol{X} \to \boldsymbol{X}} \quad \dfrac{}{f : \boldsymbol{X} \to \boldsymbol{X}, z : \boldsymbol{X} \vdash z : \boldsymbol{X}}}{f : \boldsymbol{X} \to \boldsymbol{X}, z : \boldsymbol{X} \vdash f\ z : \boldsymbol{X}}}{f : \boldsymbol{X} \to \boldsymbol{X}, z : \boldsymbol{X} \vdash f\ (f\ z) : \boldsymbol{X}}}{f : \boldsymbol{X} \to \boldsymbol{X} \vdash \lambda z.\ f\ (f\ z) : \boldsymbol{X} \to \boldsymbol{X}}}{\vdash \lambda f.\ \lambda z.\ f\ (f\ z) : (\boldsymbol{X} \to \boldsymbol{X}) \to \boldsymbol{X} \to \boldsymbol{X}}$$

Exemple 2.2.

$$\dfrac{\dfrac{\dfrac{}{a, X, t : X \to Y \vdash t : X \to Y} \quad \dfrac{}{a, X, t : X \to Y \vdash a : X}}{a, X, t : X \to Y \vdash t\ a : Y}}{a : X \vdash \lambda t.\ t\ a : (X \to Y) \to Y}\ .$$

2.2 Propriétés de la relation de typage.

Lemme 2.1 (Lemme administratif).

- ▷ Si $\Gamma \vdash M : A$ alors $\mathcal{V}\ell(M) \subseteq \mathrm{dom}(\Gamma)$.
- ▷ *Renforcement* : si $\Gamma, x : B \vdash M : A$ et $x \notin \mathcal{V}\ell(M)$ alors $\Gamma \vdash M : A$.
- ▷ *Affaiblissement* : si $\Gamma \vdash M : A$ alors, pour tout B et tout $x \notin \mathrm{dom}(\Gamma)$ alors $\Gamma, x : B \vdash A$.
- ▷ *Contraction* : si $\Gamma, x : B, y : B \vdash M : A$ alors $\Gamma, x : B \vdash M[x/y] : A$ □

Proposition 2.1 (Préservation du typage). Si $\Gamma \vdash M : A$ et $M \to_\beta M'$ alors $\Gamma \vdash M' : A$.

Preuve. On procède comme en **Théorie de la Programmation** [Chapitre 7] avec le lemme suivant.

Lemme 2.2. Si $\Gamma, x : A \vdash M : B$ et $\Gamma \vdash N : A$ alors $\Gamma \vdash M[N/x] : B$.

□

2.3 Normalisation forte.

Définition 2.5. Un λ-terme M est dit *fortement normalisant* ou *terminant* si toute suite de β-réductions issue de M conduit à une forme normale. Autrement dit, il n'y a pas de divergence issue de M.

Théorème 2.1. Si M est typage (il existe Γ, A tels que $\Gamma \vdash M : A$) alors M est fortement normalisant.

Remarque 2.1 (Quelques tentatives de preuves ratées.). ▷ Par induction sur M ? Non.

- ▷ Par induction sur la relation de typage $\Gamma \vdash M : A$? Non (le cas de l'application pose problème car deux cas de β-réductions).

Pour démontrer cela, on utilise une méthode historique : les *candidats de réductibilité*.

Définition 2.6 (Candidat de réductibilité)**.** Soit A un type simple. On associe à A un ensemble de λ-termes, noté $\mathscr{R}_A$ appelé *candidats de réductibilité* (ou simplement *candidats*) associé à A, défini par induction sur A de la manière suivante :

- ▷ $\mathscr{R}_{\boldsymbol{X}} := \{M \mid M \text{ est fortement normalisant}\}$;
- ▷ $\mathscr{R}_{A\to B} := \{M \mid \forall N \in \mathscr{R}_A, M\ N \in \mathscr{R}_B\}$.

L'idée est la suivante :

$$\begin{array}{c} M \text{ typable} \\ \Gamma \vdash M : A \end{array} \quad \rightsquigarrow \quad M \in \mathscr{R}_A \quad \rightsquigarrow \quad M \text{ fortement normalisant.}$$

Remarque 2.2 (Rappel sur le PIBF, *c.f.* **Théorie de la Programmation** [Chapitre 10])**.** Le principe d'induction bien fondé nous dit qu'une relation $\mathscr{R}$ est terminante ssi pour tout prédicat $\mathscr{P}$ sur E vérifie que si

$$\forall x \in E \left((\forall y, x \mathscr{R}\ y \implies \mathscr{P}(y)) \implies \mathscr{P}(x) \right)$$

alors $\forall x \in E, \mathscr{P}(x)$.

Proposition 2.2. Soit A un type simple. On a :

CR 1. Pour tout $M \in \mathscr{R}_A$, M est fortement normalisant.

CR 2. Pour tout $M \in \mathscr{R}_A$, si $M \to_\beta M'$ alors $M' \in \mathscr{R}_A$.

CR 3. Pour tout M *neutre* (c-à-d, M n'est pas une λ-abstraction), si $\forall M', M \to_\beta M' \implies M' \in \mathscr{R}_A$ alors $M \in \mathscr{R}_A$.

Preuve. On montre la conjonction de **CR 1**, **CR 2** et **CR 3** par induction sur A. Il y a deux cas.

- Cas $\boldsymbol{X}$ un type simple.

  CR 1. C'est vrai par définition.

  CR 2. Si M est fortement normalisant, et $M \to_\beta M'$ alors $M' \in \mathscr{R}_{\boldsymbol{X}}$.

  CR 3. Si M est neutre et si on a que « pour tout M' tel que $M \to_\beta M'$ alors $M' \in \mathscr{R}_{\boldsymbol{X}}$ » alors c'est l'induction bien fondée pour $\to_\beta$ sur $\mathscr{R}_{\boldsymbol{X}}$.

- Cas $A \to B$ un type flèche.

  CR 1. Soit $M \in \mathscr{R}_{A\to B}$. Supposons que M diverge :

  $$M \to_\beta M_1 \to_\beta M_2 \to_\beta \cdots .$$

  On a observé que $x \in \mathscr{R}_A$ pour une variable x arbitraire (conséquence de **CR 3** pour A). Par définition de $\mathscr{R}_{A\to B}$, $M\,x \in \mathscr{R}_B$. Par **CR 1** pour B, on a que $M\,x$ est fortement normalisant. Or, $M\,x \to_\beta M_1\,x$ car $M \to_\beta M_1$. On construit ainsi une divergence dans $\mathscr{R}_B$ à partir de $M\,x$:

  $$M\,x \to_\beta M_1\,x \to_\beta M_2\,x \to_\beta \cdots .$$

  C'est absurde car cela contredit que $M\,x$ fortement normalisant.

  CR 2. Soit $M \in \mathscr{R}_{A\to B}$ et $M \to M'$. Montrons que $M' \in \mathscr{R}_{A\to B}$, *i.e.* pour tout $N \in \mathscr{R}_A$ alors $M'\,N \in \mathscr{R}_B$. Soit donc $N \in \mathscr{R}_A$. On sait que $M\,N \in \mathscr{R}_B$ (car $M \in \mathscr{R}_{A\to B}$). Et comme $M \to_\beta M'$ alors $M\,N \to_\beta M'\,N$ et, par **CR 2** pour B, on a $M'\,N \in \mathscr{R}_B$. On a donc montré $\forall N \in \mathscr{R}_{A\to B}, M'N \in \mathscr{R}_B$ autrement dit, $M' \in \mathscr{R}_{A\to B}$.

  CR 3. Soit M neutre tel que $\forall M', M \to_\beta M' \implies M' \in \mathscr{R}_{A\to B}$. Montrons que $M \in \mathscr{R}_{A\to B}$. On sait que $\to_\beta$ est

terminante sur $\mathscr{R}_A$ (par **CR 1** pour A). On peut donc montrer que $\forall N \in \mathscr{R}_A$, $M\,N \in \mathscr{R}_B$ par induction bien fondée sur $\to_\beta$. On a les hypothèses suivantes :

- hypothèse 1 : pour tout M' tel que $M \to_\beta M'$ alors $M' \in \mathscr{R}_{A\to B}$;
- hypothèse d'induction bien fondée : pour tout N' tel que $N \to_\beta N'$ que $M\,N' \in \mathscr{R}_B$.

On veut montrer $M\,N \in \mathscr{R}_B$. On s'appuie sur **CR 3** pour B et cela nous ramène à montrer que, pour tout P tel que $M\,N \to_\beta P$ est $P \in \mathscr{R}_B$. On a trois cas possibles pour $M\,N \to_\beta P$.

- Si $M = \lambda x.\, M_0$ et $P = M_0[^N/_x]$ qui est exclu car M est neutre.
- Si $P = M'\,N$ alors par hypothèse 1 $M' \in \mathscr{R}_{A\to B}$ et donc $M'\,N \in \mathscr{R}_B$.
- Si $P = M\,N'$ alors, par par hypothèse d'induction bien fondée, $M\,N' \in \mathscr{R}_B$.

□

Lemme 2.3. Soit M tel que $\forall N \in \mathscr{R}_A, M[^N/_x] \in \mathscr{R}_B$. Alors $\lambda x.\, M \in \mathscr{R}_{A\to B}$.

Preuve. On procède comme pour **CR 3** pour $A \to B$. □

Lemme 2.4. Supposons $x_1 : A_1, \dots, x_k : A_k \vdash M : A$. Alors, pour tout $N_1, \dots, N_k$ tel que $N_i \in \mathscr{R}_{A_i}$, on a

$$M[^{N_1\ \cdots\ N_k}/_{x_1\ \cdots\ x_k}] \in \mathscr{R}_A.$$

On note ici la *substitution simultanée* des x_i par des N_i dans M. C'est ***n'est pas*** la composition des substitutions.

Preuve. Par induction sur la relation de typage, il y a trois cas.

- ▷ Si on a utilisé la règle de l'axiome, c'est que M est une variable : $M = x_i$ et $A = A_i$. Soit $N_i \in \mathscr{R}_{A_i}$ alors $M[{}^{N_1\ \cdots\ N_k}/_{x_1\ \cdots\ x_k}] = N_i \in \mathscr{R}_A$.
- ▷ Si on a utilisé la règle de l'application, c'est que M est une application : $M = M_1\, M_2$ et $M_1 : B \to A$ et $M_2 : B$. On a :

  $$M[{}^{N_1\ \cdots\ N_k}/_{x_1\ \cdots\ x_k}] = M_1[{}^{N_1\ \cdots\ N_k}/_{x_1\ \cdots\ x_k}]M_2[{}^{N_1\ \cdots\ N_k}/_{x_1\ \cdots\ x_k}].$$

  On conclut en appliquant les hypothèses d'inductions : $M_1[{}^{N_1\ \cdots\ N_k}/_{x_1\ \cdots\ x_k}] \in \mathscr{R}_{B\to A}$ et $M_2[{}^{N_1\ \cdots\ N_k}/_{x_1\ \cdots\ x_k}] : \mathscr{R}_B$.
- ▷ Si on a utilisé la règle de l'abstraction, c'est que $M = \lambda y.M_0$ avec $y \notin \{x_1, \ldots, x_k\} \cup \mathscr{V}\ell(N_1) \cup \cdots \cup \mathscr{V}\ell(N_k)$. Supposons que $x_1 : A_1, \ldots, x_k : A_k \vdash \lambda y.\ M_0 : A \to B$. Alors nécessairement $x_1 : A_1, \ldots, x_k : A_k, y : A \vdash M_0 : B$. Par hypothèse d'induction, on a que pour tout $N_i \in \mathscr{R}_{A_i}$ on a

  $$M_0[{}^{N_1\ \cdots\ N_k}/_{x_1\ \cdots\ x_k}][{}^{N}/_{y}] = M_0[{}^{N_1\ \cdots\ N_k\ N}/_{x_1\ \cdots\ x_k\ y}] \in \mathscr{R}_B.$$

  Par le lemme précédent, on déduit que

  $$(\lambda y.M_0)[{}^{N_1\ \cdots\ N_k}/_{x_1\ \cdots\ x_k}] = \lambda y.(M_0[{}^{N_1\ \cdots\ N_k}/_{x_1\ \cdots\ x_k}]) \in \mathscr{R}_{A\to B}.$$

□

Corollaire 2.1. Si $\Gamma \vdash M : A$ alors $M \in \mathscr{R}_A$.

2.4 Extension : le λ-calcul typé avec $\times$ et 1.

En ajoutant les couples et *unit*, il faut modifier quatre points.

Syntaxe. $M, N ::= \lambda x.\ M \mid M\ N \mid x \mid (M, N) \mid \star \mid \pi_1\ M \mid \pi_2\ M$

β-réduction.

$$\frac{M \to_\beta M'}{(M, N) \to_\beta (M', N)} \qquad \frac{N \to_\beta N'}{(M, N) \to_\beta (M, N')}$$

$$\frac{}{\pi_1\ (M, N) \to_\beta M} \quad \frac{}{\pi_2\ (M, N) \to_\beta N}.$$

Types.

$$A, B ::= X \mid A \to B \mid A \times B \mid \mathbf{1}$$

Typage.

$$\frac{}{\star : \mathbf{1}} \qquad \frac{\Gamma \vdash M : A \qquad \Gamma \vdash N : B}{\Gamma \vdash (M, N) : A \times B}$$

$$\frac{\Gamma \vdash P : M \times N}{\Gamma \vdash \pi_1\ P : M} \qquad \frac{\Gamma \vdash P : M \times N}{\Gamma \vdash \pi_2\ P : N}\ .$$

3 Introduction à la théorie de la démonstration.

3.1 Formules et preuves.

Définition 3.1. On se donne un ensemble de *variables propositionnelles*, qui seront notées X, Y, Z, *etc.* L'ensemble des *formules* est défini par la grammaire :

$$A, B ::= X \mid A \Rightarrow B.$$

Cet ensemble de formules s'appelle le « *fragment implicatif de la logique propositionnelle intuitionniste* ».

Cela peut sembler inhabituel car, généralement, on commence par introduire $\neg$, $\vee$ et $\wedge$, car on a en tête les booléens.

Définition 3.2. Les *séquents*, notés $\Gamma \vdash A$, un couple formé de Γ une ***liste*** de formules, et A une formule. La liste Γ est une *liste d'hypothèses*. On notera Γ, A la notation pour l'extension de la liste.

Définition 3.3. On *prouve* (*dérive*) les séquents à l'aides des *règles de déduction* (*d'inférence*) :

$$_{A \in \Gamma}\ \frac{}{\Gamma \vdash A}\ \mathsf{Ax} \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\ \Rightarrow_{\mathsf{I}} \qquad \frac{\Gamma \vdash A \Rightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ \Rightarrow_{\mathsf{E}}.$$

On les appelle, dans l'ordre, règle de l'*axiome*, règle de l'*introduction de* $\Rightarrow$ et règle de l'*élimination de* $\Rightarrow$. Il s'agit des *règles de déduction naturelle pour le fragment implicatif de la logique propositionnelle intuitionniste*.

Définition 3.4. Le séquent $\Gamma \vdash A$ est *prouvable* s'il existe une *preuve* (*dérivation*) ayant $\Gamma \vdash A$ à la racine et des axiomes aux feuilles. La formule A est *prouvable* si $\vdash A$ l'est.

Exemple 3.1.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{X \Rightarrow Y, X \vdash X \Rightarrow Y}\,\mathsf{Ax} \quad \dfrac{}{X \Rightarrow Y, X \vdash X}\,\mathsf{Ax}}{X \Rightarrow Y, X \vdash Y}\,\Rightarrow_{\mathsf{E}}}{X \Rightarrow Y \vdash X \Rightarrow Y}\,\Rightarrow_{\mathsf{I}}}{\vdash (X \Rightarrow Y) \Rightarrow (X \Rightarrow Y)}\,\Rightarrow_{\mathsf{I}} \quad .$$

On écrit généralement des « preuves génériques », en utilisant A, B plutôt que X, Y.

Exemple 3.2.

$$\dfrac{\dfrac{\dfrac{}{(A \Rightarrow A) \Rightarrow B \vdash (A \Rightarrow A) \Rightarrow B}\,\mathsf{Ax} \quad \dfrac{\dfrac{\dfrac{}{(A \Rightarrow A) \Rightarrow B, A \vdash A}\,\mathsf{Ax}}{(A \Rightarrow A) \Rightarrow B \vdash A \Rightarrow A}\,\Rightarrow_{\mathsf{I}}}{(A \Rightarrow A) \Rightarrow B \vdash B}\,\Rightarrow_{\mathsf{E}}}{\vdash ((A \Rightarrow A) \Rightarrow B) \Rightarrow B}\,\Rightarrow_{\mathsf{I}} \quad .$$

3.2 Et en Rocq ?

En Rocq, un objectif de preuve

$$\left.\begin{array}{l} H_1 : A_1 \\ H_2 : A_2 \\ H_3 : A_3 \\ \vdots \\ \hline\hline \end{array}\right\} \Gamma \qquad A$$

correspond au séquent

$$\Gamma \vdash A.$$

Chaque tactique correspond à des opérations sur l'arbre de preuve. On construit « au fur et à mesure » l'arbre de preuve montrant $\Gamma \vdash A$. Voici ce que quelques tactiques Rocq font.

$$\frac{??}{\Gamma, A, B, A \vdash A} \xrightarrow[\texttt{assumption}]{} \frac{}{\Gamma, A, B, A \vdash A}\,\mathsf{Ax}$$

$$\frac{??}{\Gamma \vdash C} \xrightarrow[\texttt{assert } A]{} \frac{\dfrac{??}{\Gamma, A \vdash B}}{\Gamma \vdash A \Rightarrow B} \quad \frac{??}{\Gamma \vdash A}\,\Rightarrow_{\mathsf{E}}$$

$$\frac{??}{\Gamma \vdash C} \xrightarrow[\texttt{cut } A]{} \frac{\dfrac{??}{\Gamma \vdash A \Rightarrow B} \quad \dfrac{??}{\Gamma \vdash A}}{\Gamma \vdash B}\,\Rightarrow_{\mathsf{E}}$$

$$\frac{??}{\Gamma \vdash C} \xrightarrow[\texttt{apply } H]{} \frac{\dfrac{}{\Gamma, A \Rightarrow B \vdash A \Rightarrow B}\,\mathsf{Ax} \quad \dfrac{??}{\Gamma, A \Rightarrow B \vdash A}}{\Gamma, \underbrace{A \Rightarrow B}_{H} \vdash B}\,\Rightarrow_{\mathsf{E}}$$

$$\frac{??}{\Gamma \vdash B \Rightarrow C} \xrightarrow[\texttt{intro}]{} \frac{\dfrac{??}{\Gamma, B \vdash C}}{\Gamma \vdash B \Rightarrow C}\,\Rightarrow_{\mathsf{I}}$$

3.3 Liens avec le λ-calcul simplement typé : *correspondance de Curry-Howard.*

Les règles de typage du λ-calcul correspondent au règles d'inférences du fragment implicatif :

$$\frac{}{\Gamma \vdash x : A}\ {\scriptstyle x : A \in \Gamma} \quad \longleftrightarrow \quad \frac{}{\Gamma \vdash A}\ {\scriptstyle A \in \Gamma}\ \mathsf{Ax}$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.\, M : A \to B} \quad \longleftrightarrow \quad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\ \Rightarrow_{\mathsf{I}}$$

$$\frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B} \quad \longleftrightarrow \quad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\ \Rightarrow_{\mathsf{I}}$$

.

En retirant les λ-termes en bleu (incluant les « : »), et en changeant $\to$ en $\Rightarrow$, on obtient exactement les mêmes règles.

Si on sait que $\Gamma \vdash x : A$ alors, en effaçant les parties en bleu, on obtient une preuve de $\tilde{\Gamma} \vdash A$.

Inversement, on se donne une preuve de $\Gamma \vdash A$. On se donne des variables x_i pour transformer $\Gamma = A_1, \ldots, A_k$ en $\hat{\Gamma} = x_1 : A_1, \ldots, x_k : A_k$. Par induction sur $\Gamma \vdash A$, on montre qu'il existe un λ-terme tel que $\hat{\Gamma} \vdash M : A$. On a trois cas.

- Pour $\Rightarrow_{\mathsf{I}}$, par induction, si $\hat{\Gamma}, x = A \vdash M : B$, on déduit $\hat{\Gamma} \vdash \lambda x.\, M : A \to B$.
- Pour $\Rightarrow_{\mathsf{I}}$, par induction, si $\hat{\Gamma} \vdash M : A \to B$ et $\hat{\Gamma} \vdash N : A$, on déduit $\hat{\Gamma} \vdash M\ N : B$.
- Pour Ax, on sait $A \in \Gamma$ donc il existe x tel que $x : A \in \hat{\Gamma}$, et on conclut $\hat{\Gamma} \vdash x_i : A$.

On a les propriétés suivantes pour la relation de déduction :

- *affaiblissement* : si $\Gamma \vdash B$ (implicitement « est prouvable ») alors $\Gamma, A \vdash B$;
- *contraction* : si $\Gamma, A, A \vdash B$ alors $\Gamma, A \vdash B$;
- *renforcement* si $\Gamma, A \vdash B$ alors $\Gamma \vdash B$ *à condition qu'on n'utilise pas l'axiome avec l'hypothèse* A *(celle là uniquement, les* A *intermédiaires ne posent pas de problèmes) pour déduire* B.

- ▷ *échange* ; su $\Gamma, A, B, \Gamma' \vdash C$ alors $\Gamma, B, A, \Gamma' \vdash C$.

C'est analogue aux propriétés du typage en λ-calcul.

En effet, la propriété de renforcement, très imprécise dans sa formulation logique, est simplement : si $\hat{\Gamma}, x : A \vdash M : B$ alors $\hat{\Gamma} \vdash M : B$ *à condition que* $x \notin \mathcal{V}\ell(M)$.

Si on veut prouver ces propriétés (au lieu d'utiliser la correspondance de Curry-Howard), on ferait une induction sur la preuve du séquent qui est donné.

La règle

$$\frac{\Gamma \vdash B}{\Gamma, A \vdash B}\ \mathsf{aff}$$

est *admissible*. En effet, si on sait prouver les prémisses (ici, $\Gamma \vdash B$) alors on sait prouver la conclusion (ici, $\Gamma, A \vdash B$). Ceci dépend fortement de la logique que l'on utilise.

3.4 Curry-Howard du côté calcul : les coupures.

Typons un redex :

$$\frac{\dfrac{\Gamma, x : A \vdash M : B}{\lambda x.\, M : A \to B}\ \Rightarrow_{\mathsf{I}} \qquad \Gamma \vdash N : A}{\Gamma \vdash (\lambda x.\, M)\, N : M}\ \Rightarrow_{\mathsf{E}}.$$

Oui, c'est exactement la même chose que la tactique `assert` en Rocq.

Définition 3.5. Une *coupure* est un endroit dans la preuve où il y a un usage d'une règle d'élimination ($\Rightarrow_{\mathsf{E}}$) dont la prémisse principale est déduite à l'aide d'une règle d'introduction ($\Rightarrow_{\mathsf{I}}$) pour le même connecteur logique.

Remarque 3.1. Ici, on n'a qu'un seul connecteur logique, $\Rightarrow$, mais cela s'étend aux autres connecteurs que l'on pourrait ajouter. La *prémisse principale* est, par convention, la première.

On peut *éliminer une coupure* pour $\Rightarrow$, c'est-à-dire transformer une preuve (*c.f.* contracter un β-redex) en passant de

$$\dfrac{\dfrac{\dfrac{\delta}{\Gamma, A \vdash B}}{\Gamma \vdash A \Rightarrow B}\Rightarrow_{\mathsf{I}} \qquad \dfrac{\delta'}{\Gamma \vdash A}}{\Gamma \vdash B}\Rightarrow_{\mathsf{E}}$$

à

$$\dfrac{\delta[\delta'/A]}{\Gamma \vdash B}$$

où l'on note $\delta[\delta'/A]$ la preuve obtenue en remplaçant dans δ chaque usage de l'axiome avec A par δ'.

On a le même séquent en conclusion (*c.f.* préservation du typage en λ-calcul simplement typé).

La correspondance de Curry-Howard c'est donc :

Types	$\longleftrightarrow$	Formules
Programmes	$\longleftrightarrow$	Preuves
β-réduction	$\longleftrightarrow$	Élimination d'une coupure
Programmation	$\longleftrightarrow$	***Logique***

3.5 Faux, négation, consistance.

On modifie nos formules :

$$A, B ::= X \mid A \Rightarrow B \mid \bot$$

et on ajoute la règle d'élimination du $\bot$ (il n'y a pas de règle d'introduction) :

$$\dfrac{\Gamma \vdash \bot}{\Gamma \vdash A}\bot_{\mathsf{E}}.$$

La négation $\neg A$ est une notation pour $A \Rightarrow \bot$. On peut donc prouver le séquent $\vdash A \Rightarrow \neg\neg A$:

$$\dfrac{\dfrac{\dfrac{}{A, \neg A \vdash \neg A}\,\mathsf{Ax} \qquad \dfrac{}{A, \neg A \vdash A}\,\mathsf{Ax}}{A, \neg A \vdash \bot}\,\Rightarrow_{\mathsf{E}}}{A \vdash \neg\neg A}\,\Rightarrow_{\mathsf{I}}}{\vdash A \Rightarrow \neg\neg A}\,\Rightarrow_{\mathsf{I}} \quad .$$

Théorème 3.1 (Élimination des coupures)**.** Si $\Gamma \vdash A$ (est prouvable) alors il existe une preuve ***sans coupure*** de $\Gamma \vdash A$.

Preuve. *c.f.* TD. □

Remarque 3.2 (Lien avec normalisation forte en λ-calcul simplement typé)**.** Ici, on veut la normalisation faible (« il existe une forme normale . . . »). On ne peut pas appliquer *stricto sensu* la normalisation forte pour le λ-calcul simplement typé car le système de type contient $\bot$.

Lemme 3.1. Une preuve sans coupure de $\vdash A$ en logique intuitionniste se termine (à la racine) nécessairement par une règle d'introduction.

Preuve. Par induction sur $\vdash A$. Il y a 4 cas.

- ▷ Ax : Absurde car $\Gamma = \emptyset$.
- ▷ $\Rightarrow_{\mathsf{I}}$: OK
- ▷ $\Rightarrow_{\mathsf{E}}$: On récupère une preuve de $\vdash B \Rightarrow A$ qui termine (par induction) par une introduction $\Rightarrow_{\mathsf{I}}$. Absurde car c'est une coupure.
- ▷ $\bot_{\mathsf{E}}$: On récupère une preuve de $\bot$ qui termine par une règle d'induction : impossible.

□

Corollaire 3.1 (Consistance de la logique)**.** Il n'y a pas de preuve de $\vdash$ en logique propositionnelle intuitionniste dans le fragment avec $\Rightarrow$ et $\bot$.

Preuve. S'il y en avait une, il y en aurait une sans coupure, qui se termine par une règle d'introduction, impossible. □

3.6 Et en Rocq ? (partie 2)

On étend les formules avec $\forall$, $\exists$, $\neg$, $\vee$, $\wedge$, *etc.* Les preuves sont des λ-termes. En effet, dans une preuve de $\vdash X \to X \to X$ on peut écrire

$$\texttt{exact}\ (\texttt{fun}\, x\ y \texttt{ => } x),$$

pour démontrer le séquent.

Le mot clé `Qed` prend le λ-terme construit par la preuve et calcule M' sous forme normale tel que $M \to_\beta^\star M'$. La logique de Rocq est *constructive*. C'est-à-dire qu'une preuve de $A \Rightarrow B$ c'est une fonction qui transforme une preuve de A en une preuve de B. Après avoir appelé `Qed`, il est possible d'extraire le λ-terme construit en un programme OCaml, Haskell, *etc.*

3.7 Logique intuitionniste *vs* logique classique.

Dans la logique que l'on a considérée (TD), on a deux règles d'introduction pour $\vee$:

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ \vee_{\mathsf{I}}^{\mathsf{g}} \qquad \frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ \vee_{\mathsf{I}}^{\mathsf{g}}.$$

Lorsqu'on a une preuve de $A \vee B$, on a, soit une preuve de A, soit une preuve de B. Ce n'est pas une preuve « il est impossible de ne pas avoir A et B ». La logique est *constructive*.

On rappelle que $\neg A := A \to \bot$, et que l'on se donne la règle d'élimination de $\bot$:

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash A}\ \bot_{\mathsf{E}}.$$

La *logique classique* est la logique obtenue à l'aide de l'ajout d'une des règles suivantes :

Tiers exclu. $\dfrac{}{\Gamma \vdash A \vee \neg A}$ tiers exclu

Ce n'est pas constructif : on ne sait pas si l'on a une preuve de $\Gamma \vdash A$ ou de $\Gamma \vdash \neg A$.

Absurde. $\dfrac{}{\Gamma \vdash (\neg\neg A) \Rightarrow A}$ absurde

C'est mieux : ici, on n'a pas de $\vee$.

Loi de Peirce. $\dfrac{}{\Gamma \vdash ((A \Rightarrow B) \Rightarrow A) \Rightarrow A}$ peirce

C'est encore mieux : ici, on n'a pas de $\vee$, ni de $\bot$, mais c'est plus subtil.

En choisissant un de ces axiomes, on a la même notion de prouvabilité.

Exercice 3.1. Montrons que absurde implique tiers exclu (au sens de « on peut dériver l'un dans le système incluant l'autre »).

$$\dfrac{\dfrac{}{\Gamma \vdash (\neg\neg(A \implies \neg A)) \Rightarrow (A \vee \neg A)}\,\text{absurde} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, \neg(A \vee \neg A) \vdash \neg(A \vee \neg A)}\,\text{ax} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, \neg(A \vee \neg A), A \vdash A}\,\text{ax}}{\Gamma, \neg(A \vee \neg A), A \vdash A \vee \neg A}\vee_1^g \qquad \dfrac{\dfrac{}{\Gamma, \neg(A \vee \neg A), A \vdash \neg(A \vee \neg A)}}{\Gamma, \neg(A \vee \neg A), A \vdash \bot}\Rightarrow_E}{\Gamma, \neg(A \vee \neg A) \vdash \neg A}\Rightarrow_I}{\Gamma, \neg(A \vee \neg A) \vdash A \vee \neg A}\vee_1^d}{\Gamma, \neg(A \vee \neg A) \vdash A \vee \neg A}\Rightarrow_E}{\Gamma \vdash \neg\neg(A \vee \neg A)}\Rightarrow_I}{\Gamma \vdash A \vee \neg A}\Rightarrow_E$$

Théorème 3.2 (Glivenko). Une formule A est prouvable en logique classique si et seulement si $\neg\neg A$ est prouvable en logique intuitionniste.

Preuve. Ressemble un peu à la traduction par continuation des programmes fouine. □

Corollaire 3.2. La logique classique est consistante ssi la logique intuitionniste est consistante.

Preuve. Si $\vdash \bot$ en logique intuitionniste alors $\vdash \bot$ en logique classique. Si $\vdash \bot$ en logique classique, alors $\neg\neg\bot$ en intuitionniste,

et on peut en déduire une preuve de $\vdash \bot$ en intuitionniste :

$$\dfrac{\dfrac{\text{par hyp.}}{\vdash (\bot \to \bot) \to \bot} \quad \dfrac{\dfrac{}{\bot \vdash \bot}\,\mathsf{ax}}{\vdash \bot \to \bot}\,\Rightarrow_{\mathsf{I}}}{\vdash \bot}\,\Rightarrow_{\mathsf{E}}.$$

□

3.8 Logique classique et Curry-Howard : intuition opérationnelle.

On cherche à compléter la correspondance de Curry-Howard :

Types	$\longleftrightarrow$	Formules
Programmes	$\longleftrightarrow$	Preuves
β-réduction	$\longleftrightarrow$	Élimination d'une coupure
Principes classiques	$\longleftrightarrow$	???
Programmation	$\longleftrightarrow$	***Logique***

Avec la preuve par l'absurde, on peut « recommencer dans un contexte différent ».

On recommence à montrer A dans le contexte $\Gamma', \neg A$ au lieu de $\Gamma, \neg A$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma', \neg A \vdash A}}{\Gamma', \neg A \vdash \bot}}{\Gamma', \neg A \vdash B}}{\vdots}}{\Gamma, \neg A \vdash A}}{\Gamma, \neg A \vdash \bot}}{\Gamma \vdash \neg\neg A}}{\Gamma \vdash A}}{\vdots}}{\Gamma', \neg A \vdash A}}{\Gamma', \neg A \vdash \bot}}{\Gamma' \vdash \neg\neg A}}{\Gamma' \vdash A}.$$

S'autoriser les principes classiques, c'est savoir utiliser les exceptions : si ça explose, je peux le rattraper. En effet, l'élimination du $\bot$ fait penser à un opérateur comme `raise : exn -> 'a`, et la construction `try...with...` pour pouvoir « sauter » à des endroits du programme, et dévier le flot du programme.

4 Le λ-calcul polymorphe.

On étend la grammaire des types :

$$A, B ::= X \mid A \to B \mid \forall X\ A.$$

Ici, les X ne sont plus les types de base (noté $\boldsymbol{X}$ précédemment), mais ce sont des *variables de types*.

Pour mieux refléter les notations de la littérature, on aura plutôt :

$$A, B ::= \alpha \mid A \to B \mid \forall \alpha\ A.$$

Le « $\forall \alpha\ A$ » est une structure qui lie, $\forall$ est un lieur. Il y a donc une notion de variables libres d'un type $\mathscr{Vl}(A)$, d'α-conversion, et de substitution. Par exemple,

$$\mathscr{Vl}(\forall \alpha\, \forall \beta\, (\alpha \to \beta \to \gamma \to \alpha)) = \{\gamma\}$$

et

$$\forall \alpha\ \alpha \to \alpha =_\alpha \forall \beta\ \beta \to \beta.$$

4.1 Typage, version implicite.

Aux trois règles des types simples, on ajoute les deux règles ci-dessous :

$$^{\alpha \notin \mathscr{Vl}(\Gamma)}\ \frac{\Gamma \vdash M : A}{\Gamma \vdash M : \forall \alpha\ A}\ \mathscr{T}_{\mathsf{g}} \qquad \frac{\Gamma \vdash M : \forall \alpha\ A}{\Gamma \vdash M : A[^B/_\alpha]}\ \mathscr{T}_{\mathsf{i}}.$$

Les règles correspondent à la *généralisation* et à l'*instantiation*.

Et, ***on ne change pas les λ-termes***.

4.2 Typage, version explicite.

On change les λ-termes :

$$M, N ::= x \mid M\ N \mid \lambda x : A.\ M \mid \Lambda\alpha.\ M \mid M\ A.$$

La construction $\Lambda\alpha. M$ est une *abstraction de types*, et la construction $M\ A$ est l'*application d'un terme à un type*. On note ***explicitement*** le type « d'entrée » de l'abstraction.

On change les règles de β-réduction :

$$\frac{}{(\Lambda\alpha.\ M)\ A \to_\beta M[{}^{A}/_{\alpha}]} \qquad \frac{M \to_\beta M'}{\Lambda\alpha.\ M \to_\beta \Lambda\alpha.\ M'}$$

$$\frac{M \to_\beta M'}{M\ A \to_\beta M'\ A.}$$

On change également les règles de typage :

$$\Gamma(x) = A\ \frac{}{\Gamma \vdash x : A} \qquad \frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B}$$

$$x \notin \mathrm{dom}(\Gamma)\ \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x : A.\ M : A \to B}$$

$$\alpha \notin \mathcal{V}\ell(\Gamma)\ \frac{\Gamma \vdash M : A}{\Gamma \vdash \Lambda\alpha.\ M : \forall\alpha\ A} \qquad \frac{\Gamma \vdash M : \forall\alpha\ A}{\Gamma \vdash M\ B : A[{}^{B}/_{\alpha}]}$$

Lemme 4.1. On a $\Gamma \vdash M : A$ dans le système explicite ssi il existe M' dans le système explicite avec $\Gamma \vdash M' : A$ (explicite) et M est « l'effacement » de M'.

- ▷ La représentation implicite est plus utilisée dans les langages comme OCaml, où l'on doit inférer les types.
- ▷ La représentation explicite correspond plus aux langages comme Rocq, avec un lien plus naturel avec la logique.

Exemple 4.1. Soit le λ-terme non typé $\underline{2} := \lambda f.\, \lambda z.\, f\,(f\,z)$. Comment le typer en version explicite ?

1.
$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{f : \alpha \to \alpha, z : \alpha \vdash f\,(f\,z) : \alpha}}{f : \alpha \to \alpha \vdash \lambda z : \alpha.\, f\,(f\,z) : \alpha \to \alpha}}{\emptyset \vdash \lambda f : \alpha \to \alpha.\, \lambda z : \alpha.\, f\,(f\,z) : (\alpha \to \alpha) \to \alpha \to \alpha}}{\emptyset \vdash \Lambda\alpha.\, \lambda f : \alpha \to \alpha.\, \lambda z : \alpha.\, f\,(f\,z) : \forall\alpha\, (\alpha \to \alpha) \to \alpha \to \alpha.}$$

2.
$$\dfrac{\dfrac{\dfrac{\vdots}{f : \forall\alpha\, \alpha \to \alpha, z : \beta \vdash f\,\beta\,(f\,\beta\,z) : \beta}}{f : \forall\alpha\, \alpha \to \alpha \vdash \lambda z : \beta.\, f\,\beta\,(f\,\beta\,z) : \beta \to \beta}}{\emptyset \vdash \lambda f : \forall\alpha\, \alpha \to \alpha.\, \lambda z : \beta.\, f\,\beta(f\,\beta\,z) : (\forall\alpha\, \alpha \to \alpha) \to \beta \to \beta.}$$

Exemple 4.2. On suppose que l'on a les couples. Comment compléter le séquent

$$y : \beta, z : \gamma \vdash \lambda f : \boxed{?}.\, (f\,y, f\,z) : \boxed{?}.$$

- ▷ Pour les types simples, on ne peut pas si $\beta \neq \gamma$.
- ▷ Mais, avec polymorphisme, on a :

$$y : \beta, z : \gamma \vdash \lambda f : \forall\alpha\, \alpha \to \delta.\, (f\,y, f\,z) : (\forall\alpha\, \alpha \to \delta) \to \delta \times \delta.$$

Exemple 4.3.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x : \alpha \vdash x : \alpha}}{\emptyset \vdash \lambda x.\, x : \alpha \to \alpha}}{\emptyset \vdash \lambda x.\, x : \forall \alpha\, \alpha \to \alpha}}{\emptyset \vdash \lambda x.\, x : (\forall \beta\, \beta) \to (\forall \delta\, \delta).}$$

En effet, $(\forall \beta\, \beta) \to (\forall \beta\, \beta) =_\alpha (\forall \beta\, \beta) \to (\forall \delta\, \delta)$.

Exemple 4.4 (Ouch !).

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x : \alpha \vdash x : \alpha}}{x : \alpha \vdash x : \forall \alpha\, \alpha}}{x : \alpha \vdash x : \beta}}{\emptyset \vdash \lambda x.\, x : \alpha \to \beta}}{\emptyset \vdash \lambda x.\, x : \forall \alpha\, \forall \beta\, \alpha \to \beta.}$$

NON ! On a $\alpha \in \mathcal{V}\ell(x : \alpha)$!

4.3 Point de vue logique sur le polymorphisme, système F.

On se place du point de vue Curry-Howard. On ajoute deux règles de déduction supplémentaire :

$$\alpha \notin \mathcal{V}\ell(\Gamma)\ \dfrac{\Gamma \vdash A}{\Gamma \vdash \forall \alpha\, A}\ \forall_{\mathsf{I}} \qquad \dfrac{\Gamma \vdash \forall \alpha\, A}{\Gamma \vdash A[{}^{B}/_{\alpha}]}\ \forall_{\mathsf{E}}.$$

On a, encore, un possibilité d'éliminer les coupures : si $\alpha \notin \mathcal{V}\ell(\Gamma)$,

$$\dfrac{\dfrac{\dfrac{\delta}{\Gamma \vdash A}\ \forall_{\mathsf{I}}}{\Gamma \vdash \forall \alpha\, A}\ \forall_{\mathsf{E}}}{\Gamma \vdash A[{}^{B}/_{\alpha}]} \quad \rightsquigarrow \quad \dfrac{\delta[{}^{B}/_{\alpha}]}{\Gamma \vdash A[{}^{B}/_{\alpha}]}$$

Ceci correspond, par correspondance de Curry-Howard, a une β-réduction :

$$(\Lambda\alpha\, M)\, B \to_\beta M[^B/_\alpha].$$

On a aussi un théorème de normalisation forte.

Théorème 4.1. Si $\Gamma \vdash M : A$ en λ-calcul polymorphe, alors M est fortement normalisant.

Pour démontrer ce théorème, on utilise encore les candidats de réductibilité (il ne faut définir que $\mathscr{R}_{\forall\alpha\, A}$), mais la preuve est bien plus complexe. En effet, les types polymorphes ont un grand pouvoir expressif (*c.f.* la section suivante).

Le système que l'on a s'appelle *système F*.[1] C'est de la logique du second ordre : on peut quantifier sur les variables propositionnelles. On quantifie donc sur des ensembles de valeurs au lieu de se limiter uniquement à quantifier sur les valeurs.

4.4 Représentation des connecteurs logiques.

On peut représenter les connecteurs logiques différemment, sans les ajouter explicitement, mais uniquement avec le fragment contenant $\to$ et $\forall$.

- On peut représenter $\bot := \forall\alpha\,\alpha$. En effet, on a la correspondance suivante :
$$\frac{\Gamma \vdash \forall\alpha\,\alpha}{\Gamma \vdash A}\,\forall_{\mathsf{I}} \quad \rightsquigarrow \quad \frac{\Gamma \vdash \bot}{\Gamma \vdash A}\,\bot_{\mathsf{E}}.$$
- On peut représenter $\top := \forall\alpha\,\alpha \to \alpha$.
- On peut représenter $A \wedge B := \forall\alpha\,(A \to B \to \alpha) \to \alpha$:

1. À ne pas confondre avec *Station F*.

$$\dfrac{\dfrac{}{\Gamma, c : A \to B \to \alpha \vdash c : A \to B \to \alpha} \quad \dfrac{\Gamma \vdash M : A}{\Gamma, c : A \to B \to \alpha \vdash M : A}}{\Gamma, c : A \to B \to \alpha \vdash c\, M : B \to \alpha} \quad \dfrac{\Gamma \vdash N : B}{\Gamma, c : A \to B \to \alpha \vdash M : A}$$

$$\dfrac{\dfrac{\Gamma, c : A \to B \to \alpha \vdash c\, M\, N\, \alpha}{\Gamma \vdash \lambda c.\, c\, M\, N : (A \to B \to \alpha) \to \alpha}}{\Gamma \vdash \lambda c.\, c\, M\, N : \forall \alpha\, (A \to B \to \alpha) \to \alpha}$$

d'où l'introduction du $\wedge$. Pour l'élimination, si

$$\Gamma \vdash M : \forall \alpha\, (A \to B \to \alpha) \to \alpha,$$

alors $\Gamma \vdash M\, (\lambda x.\, \lambda y.\, x) : A$ et $\Gamma \vdash M\, (\lambda x.\, \lambda y.\, y) : B$.

4.5 Le λ-calcul polymorphe, côté programmation.

« Généricité ». On peut avoir plusieurs types en même temps. Par exemple, une fonction de tri a un type

$$\forall \alpha\, (\alpha \to \alpha \to \texttt{bool}) \to \texttt{list}\ \alpha \to \texttt{list}\ \alpha.$$

« Paramétricité ». Lorsque l'on a une fonction $f : (\forall \alpha : \alpha) \to A$, la fonction n'inspecte pas son argument. Elle ne fait que le dupliquer, le passer à d'autres fonctions, le rejeter, mais elle ne peut pas voir les données sous-jacentes. (On exclue ici les fonctions types `Obj.magic`).

Quelques conséquences sur les formes normales :

- ▷ Il n'y a qu'une seule forme normale ayant un type $\forall \alpha\, \alpha \to \alpha$.
- ▷ Il n'y a que deux formes normales ayant un type $\forall \alpha \alpha \to \alpha \to \alpha$: $\lambda u.\, \lambda v.\, u$ et $\lambda u.\, \lambda v.\, v$.

On a aussi quelques « théorèmes gratuits », comme par exemple : si $\texttt{f} : \alpha \to \beta$ est croissante vis à vis de $\texttt{ordre}_\alpha : \alpha \to \alpha \to \texttt{bool}$ et $\texttt{ordre}_\beta : \beta \to \beta \to \texttt{bool}$, alors

$$\texttt{map f (sort ordre}_\alpha\ \texttt{l)} = \texttt{sort ordre}_\beta\ \texttt{(map f l)}.$$