Lambda-calcul
et sémantique des langages de programmation

Thibaut Balabonski @ *UPSay*
https://www.lri.fr/~blsk/Semantique/
Hiver 2021

Chapitre 3 - λ-calcul simplement typé

Programme incohérent en Python

```
p = (4, 2)
return p[1][0]
```

Erreur à l'exécution

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'int' object has no attribute
           '__getitem__'
```

Programme incohérent en Caml

```
let p = (4, 2) in
fst(snd p)
```

Erreur à la compilation

```
Error: This expression has type int but an
       expression was expected of type 'a * 'b
```

λ-terme incohérent

Sémantique à grands pas

$$\dfrac{\dfrac{}{\lambda x._ \Downarrow_v \lambda x._} \qquad \dfrac{\dots}{(\lambda y.\langle y, (\lambda z.z)2\rangle)4 \Downarrow_v \langle 4, 2\rangle} \qquad \dfrac{\pi_2(\langle 4, 2\rangle) \Downarrow_v \langle ??, ??\rangle}{\pi_1(\pi_2(\langle 4, 2\rangle)) \Downarrow_v ??}}{(\lambda x.\pi_1(\pi_2(x)))\ ((\lambda y.\langle y, (\lambda z.z)2\rangle)4) \Downarrow_v ??}$$

dérivation impossible à construire

Sémantique à petits pas

$$\begin{array}{ll} & (\lambda x.\pi_1(\pi_2(x)))\ ((\lambda y.\langle y, (\lambda z.z)2\rangle)4) \\ \to_v & (\lambda x.\pi_1(\pi_2(x)))\ \langle 4, (\lambda z.z)2\rangle \\ \to_v & (\lambda x.\pi_1(\pi_2(x)))\ \langle 4, 2\rangle \\ \to_v & \pi_1(\pi_2(\langle 4, 2\rangle)) \\ \to_v & \pi_1(2) \end{array}$$

obtention d'un terme irréductible

Slogan ***Well-typed programs do not go wrong***

Lien entre une analyse statique

— les types des expressions sont cohérents

et une propriété sémantique dynamique

— le programme s'exécute sans encombres

Quelle propriété de sûreté ?

— existence d'une dérivation en sémantique à grands pas ?

— existence d'une réduction en sémantique à petits pas ?

1 Types simples

Syntaxe typée

Types

$$\begin{array}{rcll} \sigma, \tau & ::= & o & \text{types de base} \\ & | & \sigma \to \tau & \text{types fonctions} \end{array}$$

Termes

$$\begin{array}{rcll} t & ::= & x & \text{variable} \\ & | & \lambda x^{\sigma}.t & \text{abstraction typée} \\ & | & t_1\ t_2 & \text{application} \end{array}$$

Notation $\tau_n \to (\tau_{n-1} \ldots (\tau_1 \to \tau_0) \ldots)$ est écrit plus simplement $\tau_n \to \tau_{n-1} \ldots \tau_1 \to \tau_0$

Types simples, notation à la Church

Jugement de typage

$$\Gamma \vdash t : \sigma$$

le terme t est bien typé de type σ dans l'environnement Γ avec Γ : ensemble de variables typées $\{x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}\}$

$$\frac{x^{\tau} \in \Gamma}{\Gamma \vdash x : \tau} \qquad \frac{\Gamma, x^{\sigma} \vdash e : \tau}{\Gamma \vdash \lambda x^{\sigma}.e : \sigma \to \tau} \qquad \frac{\Gamma \vdash e_1 : \sigma \to \tau \qquad \Gamma \vdash e_2 : \sigma}{\Gamma \vdash e_1\ e_2 : \tau}$$

Types simples, sans annotation

Jugement de typage

$$\Gamma \vdash t : \sigma$$

le terme t est bien typé de type σ dans l'environnement Γ avec Γ : fonction des variables dans les types $\{x_1 : \sigma_1, \ldots, x_n : \sigma_n\}$

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau} \qquad \frac{\Gamma, x : \sigma \vdash e : \tau}{\Gamma \vdash \lambda x.e : \sigma \to \tau} \qquad \frac{\Gamma \vdash e_1 : \sigma \to \tau \qquad \Gamma \vdash e_2 : \sigma}{\Gamma \vdash e_1\ e_2 : \tau}$$

Exercice : exemples et contre-exemples de termes typés
Donner des dérivations de typage pour les termes suivants, ou indiquer une impossibilité.

— $\lambda x.x$
— $\lambda xy.x$
— $\lambda xyz.x(yz)$
— $\lambda x.xx$

Types étendus : entiers
Nouveau type

$$\begin{array}{rcl} \sigma, \tau & ::= & \ldots \\ & | & \mathsf{int} \end{array}$$

Nouvelles règles de typage

$$\frac{}{\Gamma \vdash n : \mathsf{int}} \qquad \frac{\Gamma \vdash t_1 : \mathsf{int} \qquad \Gamma \vdash t_2 : \mathsf{int}}{\Gamma \vdash t_1 \oplus t_2 : \mathsf{int}}$$

Types étendus : booléens
Nouveau type

$$\begin{array}{rcl} \sigma, \tau & ::= & \ldots \\ & | & \mathsf{bool} \end{array}$$

Nouvelles règles de typage

$$\frac{}{\Gamma \vdash \mathsf{T} : \mathsf{bool}} \qquad \frac{}{\Gamma \vdash \mathsf{F} : \mathsf{bool}}$$

$$\frac{\Gamma \vdash t : \mathsf{int}}{\Gamma \vdash \mathsf{isZero}(t) : \mathsf{bool}}$$

$$\frac{\Gamma \vdash t_1 : \mathsf{bool} \qquad \Gamma \vdash t_2 : \tau \qquad \Gamma \vdash t_3 : \tau}{\Gamma \vdash \mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3 : \tau}$$

Types étendus : produits
Nouveau type

$$\begin{array}{rcl} \sigma, \tau & ::= & \ldots \\ & | & \tau_1 \times \tau_2 \end{array}$$

Nouvelles règles de typage

$$\frac{\Gamma \vdash t_1 : \tau_1 \qquad \Gamma \vdash t_2 : \tau_2}{\Gamma \vdash \langle t_1, t_2 \rangle : \tau_1 \times \tau_2}$$

$$\frac{\Gamma \vdash t : \tau_1 \times \tau_2}{\Gamma \vdash \pi_1(t) : \tau_1} \qquad \frac{\Gamma \vdash t : \tau_1 \times \tau_2}{\Gamma \vdash \pi_2(t) : \tau_2}$$

Types étendus : récursion
Nouvelle règle de typage

$$\frac{\Gamma \vdash t : (\sigma \to \tau) \to (\sigma \to \tau)}{\Gamma \vdash \mathsf{Fix}(t) : \sigma \to \tau}$$

2 Préservation du typage

Préservation du typage en λ-calcul pur
En sémantique à grands pas

$$\text{Si} \quad \Gamma \vdash t : \tau \quad \text{et} \quad t \Downarrow v \quad \text{alors} \quad \Gamma \vdash v : \tau$$

En sémantique à petits pas

$$\text{Si} \quad \Gamma \vdash t : \tau \quad \text{et} \quad t \to t' \quad \text{alors} \quad \Gamma \vdash t' : \tau$$

Préservation des types par β-réduction
Si $\Gamma \vdash t : \tau$ et $t \to_\beta t'$ alors

$$\Gamma \vdash t' : \tau$$

Preuve par récurrence sur $t \to_\beta t'$.

— Cas $t_1 t_2 \to t'_1 t_2$ avec $t_1 \to t'_1$
Par inversion de l'hypothèse $\Gamma \vdash t_1 t_2 : \tau$ il existe σ tel que $\Gamma \vdash t_1 : \sigma \to \tau$ et $\Gamma \vdash t_2 : \sigma$
Par hypothèse de récurrence $\Gamma \vdash t'_1 : \sigma \to \tau$ et on conclut avec la règle de typage de l'application

$$\frac{\Gamma \vdash t'_1 : \sigma \to \tau \qquad \Gamma \vdash t_2 : \sigma}{\Gamma \vdash t'_1 t_2 : \tau}$$

— Cas $t_1 t_2 \to t_1 t'_2$ avec $t_2 \to t'_2$ similaire
— Cas $\lambda x.t_0 \to \lambda x.t'_0$ avec $t_0 \to t'_0$ similaire
— Cas $(\lambda x.t_1)t_2 \to t_1\{x \leftarrow t_2\}$
Par inversion de l'hypothèse $\Gamma \vdash (\lambda x.t_1)t_2 : \tau$ il existe σ tel que $\Gamma \vdash \lambda x.t_1 : \sigma \to \tau$ et $\Gamma \vdash t_2 : \sigma$
et par inversion de $\Gamma \vdash \lambda x.t_1 : \sigma \to \tau$ on obtient $\Gamma, x : \sigma \vdash t_1 : \tau$
Reste à combiner $\Gamma, x : \sigma \vdash t_1 : \tau$ *et* $\Gamma \vdash t_2 : \sigma$ *pour parler du type de* $t_1\{x \leftarrow t_2\}$
Lemme de préservation des types par substitution
Si $\Gamma, x : \sigma \vdash t : \tau$ et $\Gamma \vdash u : \sigma$ alors $\Gamma \vdash t\{x \leftarrow u\} : \tau$

Préservation des types par substitution
Si $\Gamma, x : \sigma \vdash t : \tau$ et $\Gamma \vdash u : \sigma$ alors

$$\Gamma \vdash t\{x \leftarrow u\} : \tau$$

Preuve par récurrence sur la dérivation de $\Gamma, x : \sigma \vdash t : \tau$

— Cas où t est une variable

  — Cas $\Gamma, x : \sigma \vdash x : \tau$ avec $\sigma = \tau$
  Alors $x\{x \leftarrow u\} = u$ et $\Gamma \vdash u : \sigma = \tau$

  — Cas $\Gamma, x : \sigma \vdash y : \tau$ avec $x \neq y$ et $\Gamma(y) = \tau$
  Alors $y\{x \leftarrow u\} = y$ et $\Gamma \vdash y : \tau$

— Cas $\Gamma, x : \sigma \vdash t_1\ t_2 : \tau$ avec $\Gamma, x : \sigma \vdash t_1 : \sigma \to \tau$ et $\Gamma, x : \sigma \vdash t_2 : \sigma$
et hypothèses de récurrence $\Gamma \vdash t_1\{x \leftarrow u\} : \sigma \to \tau$ et $\Gamma \vdash t_2\{x \leftarrow u\} : \sigma$
Donc $\Gamma \vdash (t_1\{x \leftarrow u\})\ (t_2\{x \leftarrow u\}) : \tau$, ce qui permet de conclure car $(t_1\{x \leftarrow u\})\ (t_2\{x \leftarrow u\}) = (t_1\ t_2)\{x \leftarrow u\}$

— Cas $\Gamma, x : \sigma \vdash \lambda y^{\tau'}.t : \tau' \to \tau$ avec $\Gamma, x : \sigma, y : \tau' \vdash t : \tau$
et hypothèse de récurrence $\Gamma, y : \tau' \vdash t\{x \leftarrow u\} : \tau$
Donc $\Gamma \vdash \lambda y^{\tau'}.(t\{x \leftarrow u\}) : \tau$
Par α-renommage on suppose que $y \neq x$ et $y \notin \mathsf{fv}(u)$, donc $(\lambda y^{\tau'}.t)\{x \leftarrow u\} = \lambda y^{\tau'}.(t\{x \leftarrow u\})$, et le jugement précédent permet donc de conclure

Préservation des types par réduction
Conséquences

— Si un terme a un type, il n'en sortira pas par β-réduction

— Si un terme a un type et une forme normale, la forme normale a le même type

3 Sûreté du typage

Notion de sûreté
L'évaluation ne doit jamais voir d'opération incohérente
En sémantique à petits pas

— réduction jamais bloquée avant d'avoir atteint une valeur

C'est déjà exprimable : *si t n'est pas une valeur alors il existe t' tel que $t \to t'$*
En sémantique à grands pas

— la dérivation n'est jamais bloquée

Pour le formaliser : *inclure les blocages dans la sémantique*

Sûreté du typage
Lemme de progression

Si $\vdash t : \tau$ et t n'est pas une valeur alors il existe t' tel que $t \to t'$

Combinaison avec le lemme de préservation du typage on obtient $\vdash t' : \tau$, et on peut donc continuer
Théorème de sûreté
Si $\vdash t : \tau$, alors

- soit il existe $t \to t_1 \to \ldots \to t_n$ avec t_n une valeur
- soit il existe une réduction infinie $t \to t_1 \to t_2 \to \ldots$

Lemme de progression pour λ-calcul + paires (appel par valeur)
On prouve la propriété

Si $\vdash t : \tau$ alors t est une valeur ou il existe t' avec $t \to_v t'$

par récurrence sur la dérivation de $\vdash t : \tau$

- Cas $\Gamma \vdash x : \tau$ avec $\Gamma(x) = \tau$
  Impossible car on ne considère que l'environnement vide
- Cas $\vdash \lambda x.t_0 : \sigma \to \tau$ avec $x : \sigma \vdash t_0 : \tau$
  Alors $t = \lambda x.t_0$ est une valeur
- Cas $\vdash \langle t_1, t_2 \rangle : \tau_1 \times \tau_2$ avec $\vdash t_1 : \tau_1$ et $\vdash t_1 : \tau_2$
  Par hypothèse de récurrence sur $\vdash t_1 : \tau_1$ on a :
  - soit il existe t'_1 avec $t_1 \to t'_1$ et alors $\langle t_1, t_2 \rangle \to_v \langle t'_1, t_2 \rangle$
  - soit t_1 est une valeur v_1
    alors par hypothèse de récurrence sur $\vdash t_2 : \tau_2$ on a :
    - soit il existe t'_2 avec $t_2 \to t'_2$ et alors $\langle v_1, t_2 \rangle \to_v \langle v_1, t'_2 \rangle$
    - soit t_2 est une valeur v_2 et alors $\langle v_1, v_2 \rangle$ est une valeur
- Cas $\vdash t_1 t_2 : \tau$ avec $\vdash t_1 : \sigma \to \tau$ et $\vdash t_2 : \sigma$
  Comme au cas précédent :
  - soit il existe t'_1 avec $t_1 \to t'_1$, et alors $t_1 t_2 \to_v t'_1 t_2$
  - soit t_1 est une valeur v_1, et dans ce cas
    - soit il existe t'_2 avec $t_2 \to t'_2$, et alors $v_1 t_2 \to_v v_1 t'_2$
    - soit t_2 est une valeur v_2 On veut alors démontrer que $v_1\ v_2$ se réduit
      Lemme de classification : si a est une valeur et $\Gamma \vdash t : \sigma \to \tau$ alors a est de la forme $\lambda x.a'$
      Par lemme de classification, il existe x, t'_1 tels que $v_1 = \lambda x.t'_1$ et alors $(\lambda x.t'_1)v_2 \to_v t'_1\{x \leftarrow v_2\}$
- Cas $\vdash \pi_1(t_0) : \tau_1$ avec $\vdash t_0 : \tau_1 \times \tau_2$ Par hypothèse de récurrence on a :
  - soit il existe t'_0 avec $t_0 \to t'_0$, et alors $\pi_1(t_0) \to_v \pi_1(t'_0)$
  - soit t_0 est une valeur v_0, et on veut montrer que $\pi_1(v_0)$ se réduit
    Lemme de classification : si a est une valeur et $\Gamma \vdash a : \tau_1 \times \tau_2$ alors a est de la forme $\langle a_1, a_2 \rangle$
    Par lemme de classification il existe v_1, v_2 tels que $v_0 = \langle v_1, v_2 \rangle$ et alors $\pi_1(\langle v_1, v_2 \rangle) \to_v v_1$
- Cas $\vdash \pi_2(t_0) : \tau_2$ avec $\vdash t_0 : \tau_1 \times \tau_2$ identique

4 Correspondance de Curry-Howard

Programmes = preuves ***bande annonce***

λ-calcul typé	Déduction naturelle
$\dfrac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau}$	$\dfrac{\tau \in \Gamma}{\Gamma \vdash \tau}$
$\dfrac{\Gamma, x : \sigma \vdash e : \tau}{\Gamma \vdash \lambda x.e : \sigma \to \tau}$	$\dfrac{\Gamma, \sigma \vdash \tau}{\Gamma \vdash \sigma \Rightarrow \tau}$
$\dfrac{\Gamma \vdash e_1 : \sigma \to \tau \qquad \Gamma \vdash e_2 : \sigma}{\Gamma \vdash e_1\, e_2 : \tau}$	$\dfrac{\Gamma \vdash \sigma \Rightarrow \tau \qquad \Gamma \vdash \sigma}{\Gamma \vdash \tau}$
τ : type	τ : formule
$\vdash$: bon typage	$\vdash$: prouvabilité

Correspondance à la base d'assistants à la preuve

5 Normalisation

Normalisation
La « réduction » réduit-elle quelque chose ?
Théorème

Si $\Gamma \vdash t : \tau$ alors t est fortement normalisable.

Théorème de normalisation, preuve syntaxique ?
Si $\Gamma \vdash t : \tau$ alors t est fortement normalisable.
Tentative de preuve par récurrence

— Cas de la variable : x est fortement normalisable
— Cas de l'abstraction : si t_0 est fortement normalisable, alors $\lambda x.t_0$ est fortement normalisable
— Cas de l'application : si t_1 et t_2 sont fortement normalisables, que faire du scénario suivant ?

$$t_1 t_2 \to_\beta^* (\lambda x.t_1')t_2 \to_\beta^* (\lambda x.t_1')t_2' \to_\beta t_1'\{x \leftarrow t_2'\} \to_\beta ???$$

Difficulté : $t_1'\{x \leftarrow t_2'\}$ n'est pas un sous-terme de t

Lemme

Si t et u sont typés et fortement normalisables alors $t\{x \leftarrow u\}$ est fortement normalisable

Exercice : préservation de la normalisation par réduction
Si t est fortement normalisable et $t \to^* t'$ alors t' est fortement normalisable
Si t est faiblement normalisable et $t \to^* t'$ alors t' est faiblement normalisable
Si s et t sont fortement normalisables mais pas st alors il existe x, s' tels que $s \to^* \lambda x.s'$ et $s'\{x \leftarrow t\}$ n'est pas fortement normalisable

Lemme de l'application
Lemme

Si s, t et $\vec{u}$ sont fortement normalisables mais pas $st\vec{u}$, alors il existe x, s' tels que $s \to^* \lambda x.s'$ et $s'\{x \leftarrow t\}\vec{u}$ n'est pas fortement normalisable

Reformulation avec contraposée Si

— s, t et $\vec{u}$ sont fortement normalisables
— $s \to^* \lambda x.s'$
— $s'\{x \leftarrow t\}\vec{u}$ est fortement normalisable

alors $st\vec{u}$ est fortement normalisable

Ordre bien fondé
Ordre $(E, \leq)$: relation binaire $\leq$ sur l'ensemble E

— réflexive $\forall x \in E,\ x \leq x$
— anti-symétrique $\forall x, y \in E,\ x \leq y \wedge y \leq x \Rightarrow x = y$
— transitive $\forall x, y, z \in E,\ x \leq y \wedge y \leq z \Rightarrow x \leq z$

Ordre strict associé

$$x < y \quad \Longleftrightarrow x \leq y \wedge x \neq y$$

Ordre bien fondé : pas de chaîne infinie strictement décroissante

$$x_0 > x_1 > x_2 > \ldots$$

Caractérisation alternative : tout sous-ensemble non vide de E a un élément minimal

Récurrence bien fondée
Contexte : ordre bien fondé $(E, \leq)$
Pour tout prédicat P sur E

$$(\forall x \in E, (\forall y \in E, y < x \Rightarrow P(y)) \Rightarrow P(x)) \Rightarrow \forall x \in E, P(x)$$

Utilisation pour démontrer $\forall x \in E, P(x)$
Soit $x \in E$

— on suppose $P(y)$ vraie pour tout $y < x$ (hypothèses de récurrence)
— on chercher à démontrer $P(x)$

Question : y a-t-il un cas de base à cette récurrence ?

Ordre lexicographique
Produit lexicographique de deux ordres $(A, \leq_A)$ et $(B, \leq_B)$: ordre sur $A \times B$ défini par

$$(a, b) \leq (a', b') \quad \Longleftrightarrow \quad a <_A a' \vee (a = a' \wedge b \leq_B b')$$

Proposition

le produit lexicographique de deux ordres bien fondés est encore un ordre bien fondé

Conséquence : on peut faire une récurrence sur un ordre lexicographique

Exercice : terminaison de la fonction d'Ackermann
La fonction d'Ackermann est définie par les équations suivantes

$$\begin{array}{rcl} \mathsf{ack}(0,n) & = & n+1 \\ \mathsf{ack}(m+1,0) & = & \mathsf{ack}(m,1) \\ \mathsf{ack}(m+1,n+1) & = & \mathsf{ack}(m,\mathsf{ack}(m+1,n)) \end{array}$$

Démontrer que $\mathsf{ack}(m,n)$ est bien définie pour tous $m,n \in \mathbb{N}$

Lemme de préservation de la normalisation par substitution
Lemme

Si t et u sont typés et fortement normalisables alors $t\{x \leftarrow u\}$ est fortement normalisable

Par récurrence sur le produit lexicographique

$$(\mathsf{tl}(\mathsf{ty}(u)), \mathsf{ht}(t), \mathsf{tl}(t))$$

où

- $\mathsf{ty}(u)$ est le type de u
- $\mathsf{tl}(a)$ est le nombre de symboles dans a
- $\mathsf{ht}(t)$ est la longueur de la plus longue séquence de réduction à partir de t

Preuve du lemme de préservation
Raisonnement par cas sur la forme de t

- Cas de la variable
  - Cas $t = x$ alors $x\{x \leftarrow u\} = u$, fortement normalisable par hypothèse
  - Cas $t = y$ avec $y \neq x$ alors $y\{x \leftarrow u\} = y$, fortement normalisable
- Cas de l'abstraction : $t = \lambda x.t_0$
  Alors $\mathsf{ht}(t_0) = \mathsf{ht}(t)$ et $\mathsf{tl}(t_0) < \mathsf{tl}(t)$ et donc $(\mathsf{tl}(\mathsf{ty}(u)), \mathsf{ht}(t_0), \mathsf{tl}(t_0)) < (\mathsf{tl}(\mathsf{ty}(u)), \mathsf{ht}(t), \mathsf{tl}(t))$
  Donc par hypothèse de récurrence, $t_0\{x \leftarrow u\}$ est fortement normalisable
  Donc $t\{x \leftarrow u\} = \lambda x.(t_0\{x \leftarrow u\})$ est fortement normalisable.
- Cas de l'application : $t = t_0 t_1 t_2 \ldots t_n$ avec t_0 pas une application
  Par cas sur t_0
  - Cas $t_0 = y$ avec $y \neq x$
    Les réductions de t sont dans les t_i avec $i \geq 1$ donc $\mathsf{ht}(t_i) < \mathsf{ht}(t)$ pour tout $i \geq 1$
    Par hypothèse de récurrence $t_i\{x \leftarrow u\}$ est fortement normalisable pour tout $i \geq 1$
    Donc $y\ t_1\{x \leftarrow u\} \ldots t_n\{x \leftarrow u\}$ est fortement normalisable
    Donc $t\{x \leftarrow u\}$ est fortement normalisable

— Cas $t_0 = \lambda y.t'_0$
Alors $t \to t' = t'_0\{y \leftarrow t_1\}t_2 \ldots t_n$ et $\mathsf{ht}(t') < \mathsf{ht}(t)$
Donc par hypothèse de récurrence $t'\{x \leftarrow u\}$ est fortement normalisable
On a

$$\begin{array}{ll} & t'\{x \leftarrow u\} \\ = & (t'_0\{y \leftarrow t_1\}t_2 \ldots t_n)\{x \leftarrow u\} \\ = & t'_0\{x \leftarrow u\}\{y \leftarrow t_1\{x \leftarrow u\}\}\ t_2\{x \leftarrow u\}\ \ldots\ t_n\{x \leftarrow u\} \end{array}$$

Par hypothèse de récurrence $t_i\{x \leftarrow u\}$ est fortement normalisable pour tout i
Donc par lemme de l'application $t\{x \leftarrow u\}$ est fortement normalisable

— Cas $t_0 = x$ On veut montrer que $u\ t_1\{x \leftarrow u\}\ \ldots\ t_n\{x \leftarrow u\}$ Si $u \to^* y$ alors on peut conclure comme précédemment
Sinon $u \to^* \lambda y.u_0$
Par hypothèse de récurrence $t_i\{x \leftarrow u\}$ est fortement normalisable pour tout $i \geq 1$
Pour appliquer le lemme, il suffit de montrer que $t' = u_0\{y \leftarrow t_1\{x \leftarrow u\}\}\ t_2\{x \leftarrow u\}\ \ldots\ t_n\{x \leftarrow u\}$ est fortement normalisable
Montrer que $t' = u_0\{y \leftarrow t_1\{x \leftarrow u\}\}\ t_2\{x \leftarrow u\}\ \ldots\ t_n\{x \leftarrow u\}$ est fortement normalisable
Astuce : $t' = (z\ t_2\{x \leftarrow u\}\ \ldots\ t_n\{x \leftarrow u\})\{z \leftarrow u_0\{y \leftarrow t_1\{x \leftarrow u\}\}\}$ Pour conclure par hypothèse de récurrence il suffit de vérifier que :

— $z\ t_2\{x \leftarrow u\}\ \ldots\ t_n\{x \leftarrow u\}$ est fortement normalisable
— $u_0\{y \leftarrow t_1\{x \leftarrow u\}\}$ est fortement normalisable
— $\mathsf{ty}(u_0\{y \leftarrow t_1\{x \leftarrow u\}\}) < \mathsf{ty}(u)$
— $z\ t_2\{x \leftarrow u\}\ \ldots\ t_n\{x \leftarrow u\}$ est fortement normalisable car les $t_i\{x \leftarrow u\}$ sont fortement normalisables
— On a $\mathsf{ty}(u) = \mathsf{ty}(\lambda x.u_0) = \sigma \to \tau$ et $\mathsf{ty}(t_1\{x \leftarrow u\}) = \mathsf{ty}(t_1) = \sigma < \mathsf{ty}(u)$ Comme u_0 et $t_1\{x \leftarrow u\}$ sont fortement normalisables on déduit par hypothèse de récurrence que $u_0\{y \leftarrow t_1\{x \leftarrow u\}\}$ est fortement normalisable
— On a en outre $\mathsf{ty}(u_0\{y \leftarrow t_1\{x \leftarrow u\}\}) = \mathsf{ty}(u_0) = \tau < \mathsf{ty}(u)$

Donc on peut appliquer l'hypothèse de récurrence pour assurer que t' est fortement normalisable, puis le lemme pour assurer que t est fortement normalisable.

6 Sémantique dénotationnelle

Domaines sémantiques (λ-calcul avec types de base)
Sémantique dénotationnelle

— on associe chaque λ-terme t à un objet mathématique s

la nature de s dépendant du type de t
À tout type τ on associe un ensemble de valeurs mathématiques D^τ appelé *domaine sémantique* de τ

$$\begin{array}{rcl} D^{\mathsf{bool}} & = & \mathbb{B} \\ D^{\mathsf{int}} & = & \mathbb{N} \\ D^{\sigma\to\tau} & = & (D^\sigma \to D^\tau) \end{array}$$

où $A \to B$ est l'ensemble des fonctions mathématiques de A dans B

Sémantique des termes
Traduction par récurrence sur la structure du terme

$$\begin{array}{rcl} [\![x]\!]_\rho & = & \rho(x) \\ [\![\lambda x.t_0]\!]_\rho & = & a \mapsto [\![t_0]\!]_{\rho[x\leftarrow a]} \\ [\![t_1\ t_2]\!]_\rho & = & [\![t_1]\!]_\rho\ [\![t_2]\!]_\rho \end{array}$$

où ρ est un *environnement* : une fonction des variables λ vers les valeurs sémantiques, avec $\rho[x \leftarrow a]$ définie par

$$\begin{array}{rcll} \rho[x \leftarrow a](x) & = & a & \\ \rho[x \leftarrow a](y) & = & \rho(y) & \text{si } y \neq x \end{array}$$

Note : ici x est une variable λ et a est une variable mathématique

Exemples

$$\begin{array}{rcl} [\![\lambda x.x]\!]_\rho & = & a \mapsto [\![x]\!]_{\rho[x\leftarrow a]} \\ & = & a \mapsto (\rho[x \leftarrow a])(x) \\ & = & a \mapsto a \\ [\![\lambda x.\lambda y.x]\!]_\rho & = & a \mapsto [\![\lambda y.x]\!]_{\rho[x\leftarrow a]} \\ & = & a \mapsto (b \mapsto [\![x]\!]_{\rho[x\leftarrow a][y\leftarrow b]}) \\ & = & a \mapsto (b \mapsto (\rho[x \leftarrow a][y \leftarrow b])(x)) \\ & = & a \mapsto (b \mapsto a) \end{array}$$

La réduction préserve la sémantique
Théorème

$$\text{Si } t \to t', \text{ alors } [\![t]\!]_\rho = [\![t']\!]_\rho \text{ pour tout } \rho$$

la réciproque est plus subtile

Preuve : par récurrence sur $t \to t'$

Sémantique dénotationelle étendue
La plupart des extensions vues jusqu'ici peuvent être ajoutées directement

$$
\begin{array}{rcl}
[\![\mathsf{T}]\!]_\rho & = & \mathsf{vrai} \\
[\![\mathsf{F}]\!]_\rho & = & \mathsf{faux} \\
[\![n]\!]_\rho & = & n \\
[\![t_1 \oplus t_2]\!]_\rho & = & [\![t_1]\!]_\rho + [\![t_2]\!]_\rho \\
[\![\mathsf{isZero}(t)]\!]_\rho & = & \begin{cases} \mathsf{vrai} & \text{si } [\![t]\!]_\rho = 0 \\ \mathsf{faux} & \text{si } [\![t]\!]_\rho \neq 0 \end{cases} \\
& \ldots &
\end{array}
$$

Mais que dire de $[\![\mathsf{Fix}(t)]\!]_\rho$?

Domaines de Scott ***bande annonce***
Domaines sémantiques étendus

— incluant des valeurs plus ou moins définies

— munis d'un ordre d'information *min : indéfini, max : totalement défini*

— complets : toute suite croissante admet une limite

Toutes les fonctions peuvent être rendues totales
Fonctions intéressantes : *croissantes* et *continues*

— argument mieux défini donne résultat mieux défini

— image d'une limite = limite des images

On trouve alors un point fixe sémantique pour interpréter Fix *grâce au théorème de Knaster-Tarski*

DM - *à rédiger et à me renvoyer avant le prochain cours*
Théorème : préservation de la sémantique dénotationnelle par réduction

$$\text{Si } t \to_\beta t', \text{ alors } [\![t]\!]_\rho = [\![t']\!]_\rho \text{ pour tout } \rho$$