Cours de Lambda-calcul.*

Ch. Berline

CHAPITRE 1 : Syntaxe du λ-calcul non typé.
(1. termes, substitutions, réductions).†

Table des matières

Pour les trois premières sections de ce chapitre nous suivons essentiellement la présentation du livre de Krivine [2], ce qui nous permet de renvoyer à cet ouvrage pour les démonstrations manquantes.

Les paragraphes précédés d'une étoile sont des compléments, destinés par exemple à vous aider à faire le lien avec d'autres ouvrages.

Dans ces notes $=_{def}$ et $\equiv$ désignent respectivement l'égalité définitionnelle et l'identité syntaxique.

---

*DEA de Logique et Fondements de l'Informatique (2000-2003), UFR de Mathématiques, Université de Paris 7.

†Version du 31 Janvier 2002.

Notations préliminaires sur les suites finies. Nous utiliserons la notation vectorielle pour abréger l'écriture des suites finies.
Pour tout ensemble E on notera $E^{<\omega}$ l'ensemble des suites finies d'éléments de E. Les éléments de $E^{<\omega}$ seront notés $\bar{u}, \bar{v}$ etc. ; $\bar{u}\bar{v}$ désignera la concaténation des deux suites $\bar{u}$ et $\bar{v}$. On notera $l(\bar{u}) \geq 0$ la longueur de la suite $\bar{u}$, et u_i son i-ème élément (pour $0 < i \leq l(\bar{u})$); nous noterons $\bar{u}^{-i}$ la suite $\bar{u}$ privée de son i-ème élément.
Pour tout ensemble $J \subseteq [1, l(\bar{u})]$ on notera $\bar{u}^{-J}$ la suite extraite de $\bar{u}$ obtenue en en retirant les éléments dont l'indice est dans J.
Nous utiliserons les expressions $u \in \bar{v}$ et $\bar{u} \subseteq \bar{v}$; il sera entendu que dans ce cas $\bar{v}$ (resp. $\bar{u}$) désigne l'ensemble des éléments de la suite $\bar{v}$ (resp. $\bar{u}$).
Soit R une relation binaire sur E; on notera $\bar{R}$ son extension canonique à $E^{<\omega}$, définie par $\bar{u}\bar{R}\bar{v}$ si et seulement si $l(\bar{u}) = l(\bar{v})$ et pour tout $i \leq l(\bar{u})$ on a $u_i R v_i$.
Si $l(\bar{u}) = n$ et si σ est une permutation de $[1, n]$ on notera $\sigma\bar{u}$ la suite dont le ième élément est $u_{\sigma(i)}$.

1 Les λ-termes.

On dispose d'un ensemble infini de variables, V, d'un symbole de function binaire permettant d'"appliquer" un terme à un autre, et d'un opérateur d'abstraction λ. Suivant les notations de [2], t appliqué à t' sera noté $(t)t'$ [1]

Définition 1 L'ensemble L des λ-termes *est par définition le plus petit ensemble qui contient les variables et qui est clos par application et par l'opérateur d'abstraction :*
- toute variable est un terme
- si t et t' sont des termes alors $(t)t'$ est un terme
- si t est un terme et x est une variable alors $\lambda x.t$ est un terme.

Dans $(t)t'$, le terme t' sera appelé l'*argument* de t.
λ est un opérateur de liaison $\lambda x.$ est de ce point de vue analogue à $\forall x$ ou à $\delta/\delta x$). Intuitivement le terme $\lambda x.t$ dénote la "fonction" $x \mapsto t[x]$ (t peut contenir x); il s'agit là d'une interprétation "fonctionnelle" du λ.

Notation 2 *$l(t)$ est la longueur de la suite de symboles figurant dans t.*

Raisonner par induction sur la structure d'un λ-terme est équivalent à raisonner par induction sur $l(t)$. Nous utiliserons les abréviations suivantes :

Notation 3 *$\lambda x \lambda y.t$ pour $\lambda x.\lambda y.t$.*
$\lambda\bar{x}.t$ pour $\lambda x_1 ... \lambda x_n.t$, où $\bar{x} =_{def} (x_1, ..., x_n)$ et $n \geq 0$.
$u\bar{v}$ ou $uv_1...v_n$ ou pour $(..((u)v_1)v_2...)v_n$, si $\bar{v} =_{def} (v_1, ..., v_n)$ et $n \geq 0$.
Si $n = 0$, $\lambda\bar{x}.t$ et $u\bar{v}$ denotent respectivement t et u.

[1] On trouve aussi souvent $(t\,t')$ dans la littérature sur le λ-calcul. La notation mathématique usuelle serait $t(t')$.

Remarque. Nous nous permettrons, quand nous présenterons des exemples, de rajouter des paires de parenthèses ou de crochets pour accroître la lisibilité du terme ; il s'agira alors d'une notation informelle mais nous nous efforcerons qu'elle ne soit pas ambigüe. Pour raisonner formellement sur la structure d'un terme il faut évidemment toujours revenir à la syntaxe de base.

Définition 4 *L'ensemble des* sous-termes d'un terme t *de* L *est l'ensemble des termes obtenus au cours de la construction de* t; *on le définit formellement par induction sur la structure de* t.

$S(x) =_{def} \{x\}$
$S((u)v) =_{def} \{(u)v\} \cup S(u) \cup S(v)$
$S(\lambda x.u) =_{def} \{\lambda x.u\} \cup S(u)$

En particulier $S(t)$ contient t lui-même ainsi que toutes les variables qui ont une occurrence dans t (le x de $\lambda x.y$ n'est pas un sous-terme de $\lambda x.y$).

Définition 5 *L'ensemble* $Vlib(t)$ *des variables libres de* t *se définit comme d'habitude par induction sur* t. *Les* occurrences libres et liées de la variable x *dans le terme* t *se définissent aussi de la manière habituelle* [2]. *Une* variable libre de t *est une variable qui a au moins une occurrence libre dans* t. *Une* variable liée de t *est une variable de* t *qui suit immédiatement un* λ. *On dira que la variable* x *est libre (resp. liée) dans la suite de termes* $\bar{u}$ *si elle est libre (resp. liée) dans l'un des* u_i.

Une même variable x peut donc être à la fois libre et liée dans un terme t, ne revanche chaque occurrence de x dans t est soit libre soit liée.

Définition 6 *Un* terme clos *(aussi appellé* combinateur*) est un terme qui n'a pas de variable libre.*
On désignera par L° *l'ensemble des termes clos de* L

Nous verrons bientôt que nous pouvons toujours renommer les variables liées d'un terme sans en changer ni son (ou ses) eventuel sens intuitif, ni son opérationalité. La relation d'équivalence correspondante (ou α-*équivalence*) est le point de départ du λ-calcul, et il faut en démontrer les propriétés avec soin (nous, nous renverrons en partie à [K], de manière à ce que vous n'ayiez pas trop de syntaxe à ingurgiter en cours) ; notons au passage que la gestion des variables liées quand on implémente le λ-calcul ou un langage fonctionnel en machine pose de vrais problèmes conceptuels.

Exemple 7 (variables libres et liées).

Dans $(\lambda x.x)x$ *la variable* x *est à la fois libre et liée : la première occurrence de* x *est liée et la deuxième est libre.*

$\lambda x.(\lambda x.x)x$ *est clos : la première occurrence de* x *est liée par le second* λ *et la seconde par le premier. Nous pourrons toujours éviter d'écrire des termes aussi peu lisibles.*

[2] Noter que le x de $\lambda x.$ ne compte jamais comme une occurrence de x!.

Exemple 8 (entiers de Church) : $\hat{n} \equiv \lambda f.\lambda x.\underbrace{(f)(f)...(f)}_{n\,times}x$ *représente le nième itérateur : il applique n fois successivement son premier argument au second. Les entiers de Church correspondent ainsi (aussi) à une représentation unaire des entiers (n "bâtons" pour n). Une façon de coder l'entier binaire 10110 dans le même esprit est de prendre le combinateur* $\lambda f.\lambda g.\lambda x(f)(g)(f)(f)(g)x$.

Exemple 9 (les booléens Vrai et Faux) *:* $V \equiv \lambda x.\lambda y.x$ *and* $F \equiv \lambda x.\lambda y.y$.

Exemple 10 $I \equiv \lambda x.x$, $K \equiv \lambda x.\lambda y.x$ *and* $S \equiv \lambda x.\lambda y.\lambda z.((x)z)(y)z$.

K and S sont les combinateurs "basiques", ils jouent en particulier un rôle crucial dans les traductions entre le λ*-calcul et la logique combinatoire. En utilisant les abbréviations et conventions ci-dessus on peut écrire plus simplement :* $S = \lambda x \lambda y \lambda z.(xz)(y)z$.

Exemple 11 $\omega \equiv \delta\delta$, *où* $\delta \equiv \lambda x.xx$ *est le combinateur d'auto-application.*

ω *est fréquemment utilisé pour représenter les calculs qui durent infiniment sans jamais produire de résultat, même partiel (ex : "looping") ; nous en verrons bientôt une justification.*

Exemple 12 (Le combinateur de point fixe de Curry)

$Y \equiv \lambda f.(\lambda x.f(x)x)\lambda x.f(x)x$.

Cette terminologie sera elle aussi justifiée par la suite. Les combinateurs de point fixe sont un outil très puissant, qui sera crucial pour montrer que toute fonction recursive partielle est représentable par un terme du λ*-calcul pur.*[3]

Exemple 13 (λI-termes). *Un* λI-terme *est un terme qui, tel* I *(et* S, $\hat{n}$ *pour* $n \neq 0$, ω, Y*), mais contrairement à* K*, n'a pas de sous-terme de la forme* $\lambda y.u$ *avec* $y \notin Vlib(u)$.

Comme nous le verrons plus tard, les λI*-termes ont une propriété de normalisation forte, qui les fit favoriser par Church, en dépit du fait que* F *et* $\hat{0}$ *ne sont pas des* λI*-termes.*

L'exemple très simple du terme F illustre comme le même λ-terme peut porter plusieurs significations intuitives, puisque $\lambda x.\lambda y.y$ représente simultanément la seconde projection, l'entier de Church $\hat{0}$, et le booléen $F(aux)$. Il représente aussi une preuve de $A \rightarrow (B \rightarrow B)$, mais ceci est une autre histoire (typée). Ces interprétations ne sont pas des codages arbitraires : nous verrons que le comportement opérationnel de $\lambda x.\lambda y.x$ est en accord avec ces différentes interprétations.

[3] Curry avait baptisé Y le "*the paradoxical combinator*" car il permet de représenter le paradoxe de Russel par un terme du λ-calcul (où il n'est plus un paradoxe, simplement : on ne peut plus lui associer de valeur de vérité).

2 Les deux notions de substitution.

Le traitement de la substitution n'est pas la partie la plus excitante d'un cours de λ-calcul. Mais comme c'est la base du calcul il faut “faire avec”.

Le problème c'est le phénomène de “capture de variable”. Ce phénomène est déja présent en mathématique usuelle, mais il est usuellement occulté car l'intérêt est ailleurs. Donnons un exemple :

Intuitivement, quand la formule $(G) : \forall x \exists y\,(y \neq x)$ est vraie, alors toutes ses instances $(Gt)\ : \exists y\,(y \neq t)$ sont vraies, où t est un terme quelconque du langage. Or ceci est clairement faux pour $t \equiv y$. Ce qui se passe c'est que ce nouvel y est “capturé” par $\exists y$. Pour traiter le cas “y” on est amené à renommer d'abord y dans la formule G, en utilisant un nouveau nom de variable, par exemple ”z”, ce qui revient à partir de la formule “α-équivalente” $\forall x \exists z\,(z \neq x)$.

En mathématique usuelle ce traitement, c'est-à-dire tout simplement l'usage correct des règles de quantification du calcul des prédicats, n'est jamais explicite. Mais en λ-calcul ce phénomène touche au coeur de notre sujet, et on est amené à considérer deux notions de substitutions.

2.1 La substitution simple.

Définition 14 *(informelle)* La substitution simple (ou contextuelle).
Soient $u \in L$, $\bar{v} \in L^{<\omega}$, $\bar{x} \in V^{<\omega}$, on suppose que tous les x_i sont distincts et que $l(\bar{v}) = l(\bar{x})$. Par définition $u < \bar{x} := \bar{v} >$ est le terme obtenu en remplaçant simultanément toutes les occurrences libres *de chaque x_i dans u par v_i.*

En particulier on posera $u < \bar{x} := \bar{v} >=_{def} u$ si $l(\bar{x}) = 0$.

Par la suite quand nous utiliserons la notation $u < \bar{x} := \bar{v} >$ cela sous-entendra toujours que les x_i sont distincts et que $l(\bar{v}) = l(\bar{x})$.

Exemple 15 *1.* $\omega < x := t >=_{def} \omega$
2. $(\lambda x.(x)x)(x)x < x := t >=_{def} (\lambda x.(x)x)(t)t$
3. $(\lambda y.x) < x := y >=_{def} \lambda y.y$ *(capture de variable).*

Définition 16 *(formelle). On distingue 5 cas si $l(\bar{x}) > 0$.*

si	$u \equiv x_i \in \bar{x}$	*alors*	$u < \bar{x} := \bar{v} >=_{def} v_i$
si	$u \equiv y \notin \bar{x}$	*alors*	$u < \bar{x} := \bar{v} >=_{def} y$
si	$u \equiv \lambda x_i.u'$ *avec* $x_i \in \bar{x}$	*alors*	$u < \bar{x} := \bar{v} >=_{def} \lambda x_i.u' < \bar{x}^{-i} := \bar{v}^{-i} >$
si	$u \equiv \lambda y.u'$ *avec* $y \notin \bar{x}$	*alors*	$u < \bar{x} := \bar{v} >=_{def} \lambda y.u' < \bar{x} := \bar{v} >$
si	$u \equiv (w)r$	*alors*	$u < \bar{x} := \bar{v} >=_{def} (w < \bar{x} := \bar{v} >)r < \bar{x} := \bar{v} >$

Rappelons que $\bar{x}^{-i}$ est la suite extraite de $\bar{x}$ en retirant x_i, et de même pour $\bar{v}$.

Cette notion de substitution est utilisée telle quelle pour exprimer ce qu'est une congruence sur l'algèbre des termes, et plus généralement quand on souhaite “mettre des termes en contexte”, ce qui s'avère nécessaire quand on souhaite étudier le comportement opérationnel des termes dans son sens le plus général. Cependant cette notion de substitution ne peut être la base de la notion de calcul que nous introduirons bientôt (la β-réduction) car on aboutirait à un changement “du” sens profond d'un terme au cours de son calcul, de manière analogue à ce qui se passe dans le troisième example ci-dessus, où une fonction constante devient l'identité (cas $x \neq y$).

Du point de vue de l'informatique la substitution simple correspond à l'instruction d'*affectation de variable,* qui est une instruction “impérative” et n'appartient pas à la programmation fonctionnelle (pure). Il est par ailleurs bien connu que l'affectation de variable entraîne une perte de transparence regrettable de ce point de vue (dans un même programme les occurrences d'un même x peuvent prendre des valeurs tout-à-fait différentes). On ne maîtrise plus le sens de ce que l'on fait (le sens d'une expression devient dépendant du contexte).

Avant de pouvoir définir la notion de substitution correcte il nous faut toutefois étudier les propriétés de la substitution simple.

Les lemmes suivants se montrent par induction sur $l(u)$ (en pesant à distinguer deux cas si u est une variable et si u est une abstraction.). Le second lemme dit que seules les variables de $\bar{x}$ qui sont libres dans u comptent.

Lemme 17 *$u < \bar{x} := \bar{v} > \equiv u < \sigma\bar{x} := \sigma\bar{v} >$, pour toute permutation σ de* $[1, l(\bar{x})]$.

Lemme 18 *$u < \bar{x} := \bar{v} > \equiv u < \bar{x}^{-J} := \bar{v}^{-J} >$ pour tout $J \subseteq [1, l(\bar{x})]$ tel qu'aucune variable d'indice $i \in J$ n'est libre dans u.*

Lemme 19 (substitutions simples successives) *Soient $\bar{x}$ et $\bar{y}$ deux suites de variables toutes distinctes et $\bar{v}$ et $\bar{w}$ deux suites de termes. Supposons de plus que $\bar{y} \cap Vlib(\bar{v}) = \varnothing$, alors :*
$u < \bar{x} := \bar{v} >< \bar{y} := \bar{w} > \equiv u < \bar{x} := \bar{v}, \bar{y} := \bar{w} >$ $(=_{def} u < \bar{x}\bar{y} := \bar{v}\bar{w} >)$

Contre exemple si l'hypothèse n'est pas satisfaite : prenons $u \equiv (x)y$ et $v \equiv (y)y$; alors $u < x := v >< y := w > =_{def} ((y)y)y < y := w > =_{def} ((w)w)w$ tandis que $u < x := v, y := w > =_{def} ((y)y)w$.

Lemme 20 (substitution après changement de variables libres).
Si les $\bar{y}$ n'apparaîssent pas dans u, alors :
$u < \bar{x} := \bar{y} >< \bar{y} := \bar{v} > \equiv u < \bar{x} := \bar{v} >$

Définition 21 *(contextualité 1) Une relation binaire R sur L* passe au contexte *(ou :* est contextuelle*) si elle est réflexive et si elle est compatible avec l'application et la λ-abstraction :*
1. *R est réflexive.*
2. *$tRt' \Longrightarrow \lambda x.tR\lambda x.t'$ (pour tous $x \in V$ et $t, t' \in L$)*
3. *uRu' et $vRv' \Longrightarrow (u)vR(u')v'$ (pour tous $u, v, u', v' \in L$)*

On obtient une définition équivalente en remplaçant 1. par : $x \in V \Longrightarrow xRx$.

Définition 22 *(contextualité 2) Une relation binaire R sur L est contextuelle si elle est compatible avec la sustitution simple au sens suivant : pour tout terme u et toutes suites $\bar{x}$, $\bar{v}$ et $\bar{v}'$, de même longueur et de nature adéquate, on a :*
$\bar{v}\bar{R}\bar{v}' \Longrightarrow (u < \bar{x} := \bar{v} > R\, u < \bar{x} := \bar{v}' >)$.

Lemme 23 *Les deux définitions sont équivalentes.*

Preuve. Si R satisfait la seconde définition, alors elle est réflexive (cas $l(\bar{x}) = 0$), compatible avec l'application (cas $u \equiv x_1 x_2$), et enfin compatible avec l'abstraction (cas $u \equiv \lambda x.x_1$ avec x_1 distincte de x), donc elle satisfait la première définition. La réciproque se montre par induction sur la complexité de u, en pensant à distinguer deux cas différents pour $u \in V$, et aussi pour u abstraction.

Définition 24 *R est substitutive si $uRu' \Longrightarrow (u < \bar{x} := \bar{v} > R\, u' < \bar{x} := \bar{v} >)$.*

Exemples : les relations "avoir même longueur" et "avoir les mêmes variables libres" sont contextuelles mais pas substitutives ; il en sera de même de la α-équivalence, définie dans la section suivante.

**Représentation de la substitution sauvage comme "mise en contexte".*

On construit l'ensemble des termes "à trous" du λ-calcul, $L' =_{def} L\{\,[\;]_i \;/\; i \in I\,\}$, où I est un ensemble (en général fini), en ajoutant aux règles de formation des termes la règle : $[\;]_i$ est un terme pour tout $i \in I$. Un "trou" $[\;]_i$ n'est donc rien d'autre qu'une constante de terme, ou encore une variable supplémentaire qu'on ne s'autorise pas à lier, et qui sera substituée. Si C est un terme de ce calcul et $(t_i)_{i\in I}$, une famille de termes de L, $C[t_i]_i$ est le terme obtenu en plaçant chaque t_i dans tous les trous indexés par son i. Un terme C de L' (aussi noté $C[\;]_i$) est appellé *un contexte.* $C[\bar{t}]$ est une abréviation de $C[t_i]_i$.

2.2 La α-équivalence.

Intuitivement un terme u' est α-équivalent à un terme u si il est obtenu en renommant certaines des variables liées de u (éventuellement toutes ou aucune), en choisissant des noms distincts et qui n'apparaissaient pas déja dans u ("variables fraîches"), ou qui en tout cas ne changeront pas le sens du terme. Pour raisonner on a besoin d'une définition formelle, et la suivante est la plus commode pour les démonstrations mathématiques.

*(mais évidemment pas pour les implémentations, puisque, entre autre, elle demande une infinité de vérifications).

Définition 25 *On définit la relation $u =_\alpha u'$ par induction sur la longueur de u, par les clauses suivantes :*
si $u \in V$ alors $u' =_\alpha u$ si et seulement si $u' \equiv u$
si $u \equiv (v)w$ alors $u' =_\alpha u$ ssi $u' = v'w'$ avec $v' =_\alpha v$ et $w' =_\alpha w$
si $u \equiv \lambda x.v$ alors $u =_\alpha u'$ ssi $u' \equiv \lambda x'.v'$, avec $v < x := y > =_\alpha v' < x' := y >$ ppt y,
où "ppt y" est une abréviation de "pour presque tout y" et signifie "pour toute variable y sauf (éventuellement) un nombre fini".

Exemple 26 *1. $\lambda x.\lambda y.y =_\alpha \lambda x.\lambda x.x$.*
2. $(\lambda y.\lambda x.xy)(x)z =_\alpha (\lambda y.\lambda x'.x'y)(x)z$
3. $(\lambda x.\lambda x.xx)x =_\alpha (\lambda x.\lambda y.yy)x =_\alpha (\lambda z.\lambda x.xx)x$
4. $\lambda x.y \neq_\alpha \lambda x.x$ si x et y sont des variables distinctes.

La proposition suivante se déduit facilement de la définition.

Proposition 27 *$=_\alpha$ est une relation d'équivalence, appelée la α-équivalence.*

Remarque 28 *Il est immédiat que, si $u =_\alpha u'$:*
1. u' est de même nature que u (variable, abstraction ou application),
2. u et u' ont la même longueur, les mêmes variables libres, et les mêmes occurrences de variables libres (induction sur u).
3. si u ne contient pas de λ alors $u \equiv u'$.

Vérifier à partir de la définition que deux termes sont α-équivalents est un peu pénible ; il est donc utile de faire la remarque suivante :

Lemme 29 *$\lambda x.u =_\alpha \lambda x'.u < x := x' >$ pour tout x' qui n'apparaît pas dans u.*

Preuve. Comme x' n'apparaît pas dans u on a, pour tout $y \in V$, l'identité syntaxique $u < x := y > \equiv u < x := x' >< x' := y >$ (Lemme 20), ce qui est plus que suffisant pour assurer $\lambda x.u =_\alpha \lambda x'.u < x := x' >$.

Proposition 30 *Si $u =_\alpha u'$ et si aucune variable libre de $\bar{v}$ n'est liée dans u ou u', alors :*
$u < \bar{x} := \bar{v} > =_\alpha u' < \bar{x} := \bar{v} >$.

Mais $=_\alpha$ n'est pas substitutive comme le montre l'exemple suivant où l'hypothèse de la proposition n'est pas vérifiée :
$(\lambda y.x) < x := y > =_{def} \lambda y.y$ alors que $(\lambda z.x) < x := y > =_{def} \lambda z.y$.

Preuve de la proposition. **A faire en cours.**

Les termes u et u' ont les mêmes variables libres (Remarque 28).

On peut supposer que tous les x_i sont libres dans u et u' (Lemme 18)

On montre la proposition par *induction sur u*.

Le seul cas non trivial est celui où u est une abstraction. Dans ce cas $u \equiv \lambda z.w$ et $u' \equiv \lambda z'.w'$ avec $w < z := y > =_\alpha w' < z' := y >$ pour tout $y \notin F$, où F est un ensemble fini. Remarquons que $z, z' \notin \bar{x}$ puisque nous avons supposé que toutes les variables de $\bar{x}$ avaient au moins une occurrence libre dans u et u'. Il s'ensuit que $u < \bar{x} := \bar{v} > =_{def} \lambda z.w < \bar{x} := \bar{v} >$ et $u' < \bar{x} := \bar{v} > =_{def} \lambda z'.w' < \bar{x} := \bar{v} >$. Il nous reste à montrer que, pour presque tout y on a $w < \bar{x} := \bar{v} >< z := y > =_\alpha w' < \bar{x} := \bar{v} >< z' := y >$.

Supposons $y \notin \bar{x}$, alors :
$w < \bar{x} := \bar{v} >< z := y > \equiv$
$\equiv w < \bar{x} := \bar{v}, z := y >$ (Lemme 19, $z \notin Vlib(\bar{v})$ et $z \notin \bar{x}$ par hypothèse)
$\equiv w < z := y, \bar{x} := \bar{v} >$ (Lemme 17)
$\equiv w < z := y >< \bar{x} := \bar{v} >$ (Lemme 19, car $y, z \notin \bar{x}$).
On a de même :
$w' < \bar{x} := \bar{v} >< z' := y > \equiv$
$\equiv w' < z' := y >< \bar{x} := \bar{v} >$.
Supposons $y \notin F$. On a alors :

$w < z := y >=_\alpha w' < z' := y >$

On applique alors l'hypothèse d'induction à $w < z := y >$ et $w' < z' := y >$, qui sont de longueur $l(w) = l(w') < l(u)$. On conclut que pour tout $y \notin F \cup \bar{x}$ on a :

$w < \bar{x} := \bar{v} >< z := y >=_\alpha w' < \bar{x} := \bar{v} >< z' := y >$.

Proposition 31 *La relation $=_\alpha$ passe au contexte.*

Preuve. On sait déja que $=_\alpha$ est réflexive ; par ailleurs il est immédiat qu'elle est préservée par l'application. Il nous reste à voir que $u =_\alpha u'$ implique $\lambda x.u =_\alpha \lambda x.u'$, c'est à dire que $u < x := y >=_\alpha u' < x := y >$ pour presque tout y. D'après la Proposition 30 il suffit de choisir y différent de toutes les variables liées de u et u'.

Notation 32 $\Lambda =_{def} L/ =_\alpha$ *et* $\Lambda^\circ =_{def} L^\circ / =_\alpha$.

Puisque la α-équivalence est contextuelle, les opérations $u \mapsto \lambda x.u$ et $(u, v) \mapsto uv$ sont définies sur Λ. Rappelons enfin que les notions de : variable libre, occurrence de variable libre, longueur, d'un terme ne dépendent que de sa classe de α-équivalence. Nous ne travaillerons bientôt plus que dans Λ.

**Remarque* : la α-équivalence ne s'étend pas aux termes de $L[\ ]$: on n'autorise jamais de changements de variables liées dans un $C[\ ]$. On peut le faire dans un $C[u]$, puisque $C[u] \in L$, mais le terme obtenu n'est pas nécessairement de la forme $C[u']$ avec $u' =_\alpha u$. Exemple : $C \equiv \lambda x.[\ ]$ et $u \equiv x$. En revanche il est vrai que $u =_\alpha u' \Longrightarrow C[u] =_\alpha C[u']$ (contextualité de $=_\alpha$!).

2.3 La substitution correcte.

Lemme 33 *Soit $u \in L$ et $F \subseteq V$ tel que $V - F$ est infini. Alors il existe $u' =_\alpha u$ tel qu'aucune variable de F n'est liée dans u'.*

Preuve. Induction facile sur u. **A Faire en cours.**

Le seul cas non trivial est le cas où $u := \lambda x.w$. On commence par remarquer que pour tout $x' \notin Vlib(w)$ on a $\lambda x.w =_\alpha \lambda x'.w < x := x' >$ (Lemme 29), puis on utilise l'hypothèse d'induction pour trouver w" tel que w" $=_\alpha w < x := x' >$ et tel que aucune variable liée de w" n'est dans F. Comme $=_\alpha$ est contextuelle et transitive on en déduit $\lambda x.w =_\alpha \lambda x'.w$" et comme on avait choisi $x' \notin F$ c'est fini ($u' =_{def} \lambda x'.w''$ convient).

Définition 34 La substitution correcte, *c'est-à-dire sans capture de variables, est définie ainsi :*

$u[\bar{x} := \bar{v}] =_{def} u' < \bar{x} := \bar{v} >$ *où u' est un terme quelconque α-équivalent à u et dont aucune variable liée n'est libre dans $\bar{v}$.*

Le fait qu'il existe un tel u' est assuré par le lemme précédent. La Proposition 30 nous assure que, puisque tous les u' possibles sont α-équivalents, tous les

$u' < \bar{x} := \bar{v} >$ seront aussi α-équivalents. Cette définition définit donc $u[\bar{x} := \bar{v}]$ à α-équivalence près. Enfin, comme $=_\alpha$ passe au contexte, l'application $u, \bar{v} \mapsto u[\bar{x} := \bar{v}]$ est définie sur $\Lambda^{l(\bar{v})+1}$.

En résumé, la substitution correcte est une opération bien définie, à condition de la regarder à α-équivalence près, c'est-à-dire sur Λ.

Exemple 35 *1.* $(\lambda y.x)[x := y] \ =_{def} \ (\lambda z.x) < x := y > =_{def} \ \lambda z.y$.
2. Si $\bar{v}$ est clos alors $t[\bar{x} := \bar{v}] =_{def} t < \bar{x} := \bar{v} >$

Nous allons maintenant lister les propriétés de base de la substitution correcte. Les trois premiers lemmes sont exactement analogues à ce qui se passe pour la substitution simple, et s'en déduisent facilement (**Rédiger une démo**.). Le quatrième est en revanche plus puissant que son analogue “simple”.

L'égalité sur Λ sera tout simplement notée “=”.

Lemme 36 *$u[\bar{x} := \bar{v}] = u[\sigma\bar{x} := \sigma\bar{v}]$, pour toute permutation σ de $[1, l(\bar{x})]$.*

Lemme 37 *$u[\bar{x} := \bar{v}] = u[\bar{x}^{-J} := \bar{v}^{-J}]$ pour tout $J \subseteq [1, l(\bar{x})]$ tel qu'aucune variable d'indice $i \in J$ n'est libre dans u. En particulier si u est clos alors $u[\bar{x} := \bar{v}] = u$.*

Lemme 38 (substitution après changement de variables libres).
Si les $\bar{y}$ n'apparaîssent pas dans u, alors :
$u[\bar{x} := \bar{y}][\bar{y} := \bar{v}] = u[\bar{x} := \bar{v}]$

Lemme 39 (substitutions correctes successives) *Soient $\bar{x}$ et $\bar{y}$ telles que $\bar{x} \cap \bar{y} = \emptyset$, alors pour tout u et pour tous $\bar{v}$ et $\bar{w}$ de longueur correcte on a :*
1. $u[\bar{x} := \bar{v}][\bar{y} := \bar{w}] = u[\bar{x} := \bar{v}', \bar{y} := \bar{w}]$, *où* $\bar{v}' = \bar{v}[\bar{y} := \bar{w}]$
Supposons de plus les x_i ne sont pas libres dans $\bar{w}$, alors on a :
2. $u[\bar{x} := \bar{v}][\bar{y} := \bar{w}] = u[\bar{y} := \bar{w}][\bar{x} := \bar{v}']$.

Preuve. 1. se montre par induction sur u.
2. **A faire en cours.**
$u[\bar{x} := \bar{v}][\bar{y} := \bar{w}] = u[\bar{x} := \bar{v}', \bar{y} := \bar{w}]$ (en utilisant 1)
$= u[\bar{y} := \bar{w}, \bar{x} := \bar{v}']$ (Lemme 36)
$= u[\bar{y} := \bar{w}][\bar{x} := \bar{v}']$ (par 1, compte tenu que les x_i ne sont pas libres dans $\bar{w}$).

Contre-exemple à 2. pour la substitution simple (on suppose $y \notin Vlib(w)$) :
$(\lambda x.y) < y := x >< x := w >=_{def} (\lambda x.x) < x := w >=_{def} \lambda x.x$
$(\lambda x.y) < x := w >< y := w >=_{def} (\lambda x.y) < y := w >=_{def} \lambda x.w$.

Remarque 40 *1.* $(u)u'[\bar{x} := \bar{v}] = (u[\bar{x} := \bar{v}])u'[\bar{x} := \bar{v}]$
2. Si y n'est pas libre dans $\bar{v}$ et $\bar{x}$, alors $(\lambda y.u)[\bar{x} := \bar{v}] = \lambda y.u[\bar{x} := \bar{v}]$.

3 β-réduction et β-équivalence.

Dans cette section nous définissons la β-réduction[4], qui est la notion (de base) de calcul sur les termes de Λ, ainsi que la β-équivalence, qui est la relation d'équivalence qu'elle engendre. Nous posons d'abord les fondations dans L.

3.1 Redex et réduits.

Définition 41 *On appelle* redex [5] *tout terme de L de la forme $(\lambda x.u)v$; on appelle* réduit *de ce rédex le terme $u[x := v]$ (qui est un terme de Λ).*

Intuitivement il s'agit donc du passage de l'application (fonctionnelle) potentielle à une application effective). Notons que le "réduit" peut-être beaucoup plus long que le redex dont il est issu, puisque x peut avoir plusieurs occurrences libres dans u.

Proposition 42 *Tout terme α-équivalent à un redex est un redex, et leurs réduits sont égaux dans Λ.*

Preuve. **Faire démo. en cours.** La première assertion est conséquence immédiate de la Remarque 28. Montrons la seconde.

Supposons donc que l'on a $(\lambda x.v)w =_\alpha (\lambda x'.v')w'$, le but étant de montrer $v[x := w] = v'[x' := w']$ dans Λ. L'hypothèse est équivalente à : $w =_\alpha w'$ et $v < x := y >=_\alpha v' < x' := y >$ pour tout $y \notin F$, où F est un certain ensemble fini. Soient $\tilde{v}, \tilde{v}' \in L$, respectivement α-équivalents à v et v', et ne liant respectivement aucune des variables libres de w et w'; par définition de la sustitution correcte on a :

$v[x := w] =_{def} \tilde{v} < x := w >$ et $v'[x := w'] =_{def} \tilde{v}' < x := w' >$

Choisissons une variable auxiliaire $y \notin F$ qui de plus n'apparaît pas dans $\tilde{v}$,$\tilde{v}'$, w, w'. On a :

$\tilde{v} < x := y >=_\alpha v < x := y >$ (Prop. 30, puisque $v =_\alpha \tilde{v}$ et $y \notin v, \tilde{v}'$).

De même :

$\tilde{v}' < x := y >=_\alpha v' < x := y >$. Il en résulte :

$\tilde{v} < x := y >=_\alpha \tilde{v}' < x := y >$ (hypothèse sur $v, v'.et\ y \notin F$). On en déduit :

$\tilde{v} < x := y >< y := w >=_\alpha \tilde{v}' < x := y >< y := w >$ (Prop. 30, puisque les variables libres de w ne sont pas liées dans $\tilde{v}, \tilde{v}'$). Par ailleurs

$\tilde{v}' < x := y >< y := w >=_\alpha \tilde{v}' < x := y >< y := w' >$ (Prop. 31). Donc

$\tilde{v} < x := y >< y := w >=_\alpha \tilde{v}' < x := y >< y := w' >$ et finalement

$\tilde{v} < x := w >=_\alpha \tilde{v}' < x := w' >$ (Lemme 20).

Ainsi, l'opération de réduction d'un redex est en fait une opération bien définie sur Λ.

---

[4] Dans [K] Krivine utilise "*β-conversion*" comme synonyme de "β-réduction", ce qui correspond mieux à l'intuition. Cependant, partout ailleurs, "*β-conversion*" signifie β-équivalence (cf. [Bar]). Nous éviterons donc d'employer ce terme.

[5] "expression réductible".

Définition 43 Un redex d'un terme t *(de L ou de Λ) est par définition un sous-terme de t qui est un redex. Un terme de L ou Λ est dit* normal *si il n'a pas de redex.*

Noter qu'un redex de t peut avoir plusieurs occurrences dans t ; par ailleurs deux redex d'un terme peuvent être emboîtés, comme dans $I(I)K \equiv (\lambda x.x)(\lambda y.y)K$, ou disjoints, comme dans $IK(I)K \equiv ((\lambda x.x)K)(\lambda y.y)K$, et ce sont les deux seules possibilités.

A partir de maintenant, et sauf mention explicite du contraire, on ne travaillera plus que dans Λ, c'est à dire modulo $=_\alpha$, et (rappel)**, en ce qui concerne la syntaxe, "$=$" désignera l'égalité dans** Λ (c'est-à-dire, essentiellement, la α-équivalence sur L).

Pour une relation binaire R **sur** Λ on définit le fait d'être *contextuelle* ou *substitutive* de façon analogue à ce que l'on a fait sur L, *mais* en travaillant avec la substitution correcte au lieu de la substitution sauvage.

En particulier R est *substitutive et contextuelle* si et seulement si :

$$\forall t, t', \bar{x}, \bar{v}, \bar{v}' \, [\, (tRt' \wedge \bar{v}\bar{R}\bar{v}') \Rightarrow (t[\bar{x} := \bar{v}] \, R \, t'[\bar{x} := \bar{v}'])\,]$$

3.2 La β_0-réduction.

Une étape de calcul à l'intérieur d'un terme t non normal consiste à choisir un redex arbitraire et à le remplacer par son réduit.

Notation 44 *On écrit $t \to_{\beta_0} t'$ si t' est obtenu à partir de t en effectuant une étape de réduction. Une* suite de réductions *est une suite d'étapes de réduction, par exemple : $IK(I)K \to_{\beta_0} K(I)K \to_{\beta_0} KK$.*

Exemple 45 *1. $\lambda x.\lambda y.(z)\lambda u.xuyz$, étant normal, n'a pas de β_0-réduit.*
2. $(\lambda y.\lambda x.y)x \to_{\beta_0} \lambda x.y\ [y := x] =_{def} \lambda z.y\ < y := x >=_{def} \lambda z.x$
(on choisit $z \neq x$).
3. $\omega \to_{\beta_0} \omega$
Car $\omega =_{def} \delta\delta =_{def} (\lambda x.(x)x)\lambda x.(x)x \to_{\beta_0} (x)x\,[x := \lambda x.(x)x] =_{def}$
$=_{def} (\lambda x.(x)x)\lambda x.(x)x =_{def} \omega$
4. $\delta_3\delta_3 \to_{\beta_0} \delta_3\delta_3 \to_{\beta_0} ... \to_{\beta_0} \delta_3\delta_3...\delta_3$ (n fois), où $\delta_3 =_{def} \lambda x.xxx$
5. $\lambda z.(z)(\lambda y.\lambda x.y)x \to_{\beta_0} \lambda z.(z)\lambda\nu.x$
$(\lambda z.(z)(\lambda y.\lambda x.y)x)u \to_{\beta_0} (\lambda z.(z)\lambda\nu.x)u \to_{\beta_0} U =_{def} (u)\lambda\nu.x$
$(\lambda z.(z)(\lambda y.\lambda x.y)x)u \to_{\beta_0} (u)(\lambda y.\lambda x.y) \to_{\beta_0} U.$
6. (exemple simple de création de redex).
$(\lambda z.(z)y)\lambda x.x \to_{\beta_0} (\lambda x.x)y \to_{\beta_0} y$

On dit qu'il y a *création de redex* pendant une étape de réduction, si une abstraction se retrouve dotée d'un argument, alors qu'elle n'en avait pas avant. Notons que la réduction $\omega \to_{\beta_0} \omega$ de l'exemple 3. avait elle aussi créé un redex, et que dans le dernier exemple il y a création de redex si et seulement si u est une abstraction.

Définition 46 *alternative de la β_0-réduction. On définit l'ensemble $A(t)$ des β_0-réduits de t par induction sur $l(t)$:*

1. si $t \in V$ alors $A(t) =_{def} \emptyset$
2. si $t = \lambda x.u$ alors $t' \in A(t)$ si et seulement si $t' = \lambda x.u'$ avec $u' \in A(u)$
3. si $t = (u)v$ alors $t' \in A(t)$ ssi on est dans un des 3 cas suivants :
a) $t' = (u)v'$ et $v' \in A(v)$
b) $t' = (u')v$ et $u' \in A(u)$
c) $u = \lambda x.w$ et $t' = w[x := v]$

Remarque 47 *β_0 n'est pas réflexive, mais on peut avoir $t \to_{\beta_0} t$ (ex : $t = \omega$).*
β_0 n'est évidemment pas transitive
β_0 n'est pas contextuelle, car $u' \in A(u)$ et $v' \in A(v)$ n'impliquent pas $(u')v' \in A((u)v)$: il faut deux étapes pour passer de $(u)v$ à $(u')v'$.
β_0 ne crée pas de variable libre : $t \to_{\beta_0} t'$ implique $Vlib(t') \subseteq Vlib(t)$.

Lemme 48 *β_0 est substitutive : $t \to_{\beta_0} t'$ implique $t[\bar{x} := \bar{v}] \to_{\beta_0} t'[\bar{x} := \bar{v}]$.*

Preuve. Par induction sur t, en utilisant le Lemme 39 dans le cas où $t \equiv R$ est un redex :

Si t est une variable c'est ok car la prémisse est toujours fausse.

Si $t = \lambda y.u$ on peut supposer (puisqu'on travaille à α-équivalence près) que y n'est libre ni dans $\bar{v}$ ni dans $\bar{x}$. On a alors $t' = \lambda y.u'$ avec $u \to_{\beta_0} u'$ et :

$t[\bar{x} := \bar{v}] = \lambda y.u[\bar{x} := \bar{v}] \to_{\beta_0} \lambda y.u'[\bar{x} := \bar{v}] = t'[\bar{x} := \bar{v}]$ (car y n'est pas libre dans u').

Si $t = (u)v$ on a trois cas à distinguer :
- si $t' = (u)v'$ avec $v \to_{\beta_0} v'$, on applique l'hypothèse d'induction à v.
- si $t' = (u')v$ avec $u \to_{\beta_0} u'$, on applique l'hypothèse d'induction à u.
- si $u = \lambda y.w$ et $t' = w[y := v]$. On peut supposer, comme précédemment, que y n'est pas libre dans $\bar{v}$ et ne figure pas dans $\bar{x}$, et on alors :

$t[\bar{x} := \bar{v}] = (\lambda y.w\,[\bar{x} := \bar{v}])\,v[\bar{x} := \bar{v}] \to_{\beta_0} w[\bar{x} := \bar{v}][y := v[\bar{x} := \bar{v}]]$
$= w[y := v][\bar{x} := \bar{v}] = t'[\bar{x} := \bar{v}]$, l'avant-dernière égalité étant justifiée par le Lemme 39, qui s'applique à cause des hypothèses que nous avons mises sur y.

3.3 La β-réduction.

Rappelons que la *clôture réflexive* d'une relation binaire R sur Λ est la plus petite relation binaire sur Λ qui contient R et qui est réflexive. C'est aussi l'intersection des relations binaires sur Λ qui ont cette propriété. On définit de manière analogue la *clôture transitive* R^t de R, sa *clôture réflexive et transitive* R^{rt}, etc. On remarquera que *la clôture réflexive, symétrique et transitive* R^{rst} de R est la plus petite relation d'équivalence contenant R. Il est facile de vérifier que pour tous $t, t' \in \Lambda$ on a :

$tR^{rt}t'$ ssi $\exists n \geqslant 0\ \exists\, t_0, ..., t_n$ ($t_0 = t$, $t_n = t'$ et $\forall i \in [0, n-1]$ t_iRt_{i+1})

$tR^{rst}t'$ ssi $\exists n \geqslant 0\ \exists\, t_0, ..., t_n$ ($t_0 = t$, $t_n = t'$ et $\forall i \in [0, n-1]$ (t_iRt_{i+1} *ou* $t_{i+1}Rt_i$)

Il est aussi facile de vérifier que si R est contextuelle (resp. substitutive), alors R^{rt} et R^{rst} le sont aussi.

Définition 49 *La* β-réduction, *notée* $\to_\beta$, *est la clôture réflexive et transitive de* $\to_{\beta_0}$. *On a donc :*
$t \to_\beta t'$ *ssi* $\exists n \geqslant 0\ \exists t_0, ..., t_n$ *(*$t_0 = t$, $t_n = t'$ *et* $\forall i \in [0, n-1]\ t_i \to_{\beta_0} t_{i+1}$*).*
On dira que t' *est* un β-réduit *de* t.

Lemme 50 *La* β*-réduction est contextuelle et substitutive.*

Preuve. Pour la substitutivité c'est une conséquence immédiate de la propriété analogue de la β_0-réduction. L'argument ne s'applique pas pour la contextualité, puisque β_0 n'est pas contextuelle, mais il est très facile de vérifier directement que la β-réduction est contextuelle.

Il est quasiment immédiat que :

Définition 51 *alternative. La* β-réduction *est la plus petite relation binaire transitive et contextuelle qui contient tous les couples* $((\lambda x.u)v, u[x := v])$.

Une des propriétés les plus importantes de la β-réduction est la propriété de *Church-Rosser (CR)*, ou propriété de *confluence.*

Définition 52 *Une relation binaire* R *sur un ensemble* E *est* confluente *si :*
$\forall t, t_1, t_2 \in E$ $(tRt_1$ *et* $tRt_2 \Rightarrow \exists t'\ t_1Rt'$ *et* $t_2Rt')$.

Théorème 53 ***(Church-Rosser, 1936).*** *La* β*-réduction est une relation confluente.*

Preuve. Nous la repoussons en Section 4, préférant en tirer d'abord les conséquences. L'idée est d'exhiber une relation confluente dont $\to_\beta$ est la clôture transitive, et d'appliquer le lemme suivant.

Lemme 54 *La clôture transitive d'une relation confluente est encore confluente.*

Preuve. Facile **(à faire en cours).**

3.4 Termes normaux.

Les caractérisations suivantes des termes normaux nous permettent de raisonner par induction sur leur structure. La proposition suivante est triviale.

Proposition 55 *Un terme* t *est normal si et seulement si il est obtenu en appliquant un nombre fini de fois les règles suivantes :*
a) une variable est un terme normal,
b) si t *est un terme normal, alors* $\lambda x.t$ *l'est aussi (pour tout* x*),*
c) si u *et* v *sont normaux et si* u *ne commence pas par* λ, *alors* $(u)v$ *est normal.*

Proposition 56 *Un terme* t *est normal si et seulement si il est de la forme* $\lambda\bar{x}.y\bar{t}$, *où* $y \in V$, $l(\bar{x}) \geqslant 0$, $l(\bar{t}) \geqslant 0$ *et tous les* t_i *sont normaux.*

Preuve. Le fait que les termes de cette forme soient normaux est trivial. Pour prouver la réciproque on commence par remarquer qu'il existe $\bar{x}$ telle que $l(\bar{x}) \geq 0$ et $t \equiv \lambda\bar{x}.u$ avec u ne commençant pas par λ. Soit $n \geq 1$ le plus grand entier tel que $u \equiv v_1 v_2 ... v_n$. On remarque que v_1 ne peut pas être de la forme vw, par maximalité de n, et qu'il ne peut pas non plus être une abstraction : en effet on aurait dans ce cas $n \geq 2$ (puisque u n'est pas une abstraction) et $n < 2$ puisque t ne contient pas de redex, contradiction. Donc v_1 est une variable ; par ailleurs le fait que t est normal force les autres v_i à être normaux aussi.

Remarque 57 *Si t est normal tout β-réduit de t est égal à t, mais la réciproque est fausse (ex : $t = \omega$).*

3.5 Termes normalisables.

Définition 58 *Un terme u est* normalisable *si il existe un terme normal n tel que $u \to_\beta n$. Une* suite de normalisation *est une suite de β_0-réductions qui se termine sur un terme normal. Un terme est* fortement normalisable *si il n'existe aucune suite infinie de β_0-réductions partant de u.*

La proposition suivante est un corollaire immédiat du Théorème de Church-Rosser.

Proposition 59 *Si t est normalisable, alors toutes ses suites de normalisation terminent sur le même terme normal.*

L'importance du théorème de Church-Rosser, en ce qui concerne le calcul est le suivant. Soient deux algorithmes de réduction qui, partant d'un terme, choisissent à chaque étape le redex à réduire en suivant une stratégie fixée, le réduisent, et s'arrètent uniquement quand ils rencontrent un terme normal (en rendant ce terme). Si on donne à ces algorithmes un même terme t et si ils terminent tous deux le calcul, alors on sait que le résultat sera indépendant de l'algorithme (de la stratégie). Nous verrons dans ce cours qu'il existe une stratégie "complète", en ce sens que pour tout terme t normalisable le calcul effectué en suivant cette stratégie s'arrète (sur la forme normale de t) : il s'agit de la "réduction gauche" qui réduit systématiquement le redex dont le "(λ" est le plus à gauche dans le terme. Ce n'est pas le cas de toutes les stratégies : il existe même des stratégies qui ne finissent que sur les termes fortement normalisables (stratégies "perpétuelles" [Bar]).

Tout terme fortement normalisable (et en particulier tout terme normal) est évidemment normalisable. La réciproque est vraie pour les λI-termes, comme nous le montrerons dans un chapitre suivant :

Théorème 60 *Pour les λI-termes : normalisable $\iff$ fortement normalisable.*

3.6 La β-équivalence.

Définition 61 *La* β-équivalence*, notée ici* $=_\beta$*, est la plus petite relation d'équivalence sur* Λ *qui contienne* $\rightarrow_{\beta_0}$ *. On a donc :*

$t =_\beta t'$ *ssi il existe un* zigzag *entre* t *et* t'*, c'est à dire ssi :*

$\exists n\ \exists t_0, ..., t_n\ (t_0 = t,\ t_n = t'\ \text{ et } \forall i \in [0, n-1]\ (t_i \rightarrow_\beta t_{i+1}\ \text{ou}\ t_{i+1} \rightarrow_\beta t_i))$

Définition 62 *(alternative) La* β*-équivalence est la clôture symétrique et transitive de* $\rightarrow_\beta$*.*

Il est facile de vérifier que ces deux définitions sont équivalentes.

Il est important de remarquer que, contrairement à ce qui se passe pour $\rightarrow_{\beta_0}$ (et donc pour $\rightarrow_\beta$), il n'y a pas de relation entre les variables libres de deux termes β-équivalents : $Fx =_\beta I =_\beta Fy$, pour $F = \lambda x.\lambda y.y$.

Exemple 63 (Le combinateur de point fixe de Curry) *:*
Soit $Y \equiv \lambda f.(\lambda x.f(x)x)\,\lambda x.f(x)x$ *alors, pour tout terme* u*, on a* $Yu =_\beta u(Y)u$*. Autrement dit tout terme* u *a (au moins) un point fixe dans* $\Lambda/=_\beta$*, à savoir* Yu *. Noter aussi que* $\omega =_\beta YI$*.*

Proposition 64 *La* β*-équivalence est contextuelle et substitutive.*

C'est une conséquence immédiate des propriétés analogues de la β-réduction.

Nous donnons maintenant une série de conséquences triviales mais importantes du Théorème de Church-Rosser (Théorème 53).

Proposition 65 *Deux termes sont* β*-équivalents si et seulement si ils ont un réduit commun.*

Preuve. Par induction sur la longueur du zigzag entre t et t' : si $n = 1$ c'est trivial, sinon l'hypothèse d'induction dit que t et t_{n-1} ont un β-réduit commun et, selon la direction de la flèche qui existe entre t_{n-1} et t_n on conclut soit avec la transitivité de $\rightarrow_\beta$ seule soit avec le Théorème de Church Rosser et la transitivité.

Corollaire 66 *1. Deux termes normaux distincts ne sont jamais* β*-équivalents.*
2. Un terme normalisable n'est jamais β*-équivalent à un terme non normalisable.*

Corollaire 67 $\Lambda/=_\beta$ *et* $\Lambda^\circ/=_\beta$ *sont infini.*

Ce dernier résultat implique *la consistance du* λ*-calcul*, en ce sens que la β-équivalence n'identifie pas tous les termes. Il est conséquence triviale du fait qu'il existe une infinité de termes normaux clos, par exemple les entiers de Church.

La proposition 64 nous dit en particulier que les opérations d'application et d'abstraction, sont compatibles avec la β-équivalence et peuvent donc être définies sur le quotient. On obtient ainsi le "*modèle des termes du* λ*-calcul*", dont l'ensemble de base est $\Lambda/=_\beta$. Pendant plus de 30 ans ce modèle syntactique (et sa variante extensionnelle que nous verrons plus loin) est resté le seul connu pour le λ-calcul : le premier modèle non syntaxique fut construit par Scott en 1968, et obtenu comme limite inverse d'une famille de treillis complets, dans une catégorie adéquate ; nous en verrons une construction non catégorique plus simple dans la suite de ce cours.

Définition 68 *On appellera* terme essentiellement clos *tout terme β-equivalent à un terme clos.*

Par exemple : Fx est un terme essentiellement clos.

Exercice 69 *Soit t essentiellement clos et t' tel que $t \to_\beta t'$. Alors t' est essentiellement clos, si de plus t' est normal alors t' est clos. Plus généralement un terme essentiellement clos est un terme qui se β-réduit à un terme clos.*

3.7 La η- et la $\beta\eta$-equivalences.

Idée de base. Tout terme t a le même comportement applicatif que le terme $\lambda x.tx$ pour $x \notin Vlib(t)$, en ce sens que pour tout terme u on a $(\lambda x.tx)u =_\beta tu$. Par ailleurs si t n'est pas β-équivalent à une abstraction (ex : $t \in V$) il est clair que t et $\lambda x.tx$ ne sont pas β-équivalents ; il existe donc des éléments de $\Lambda/=_\beta$ qui sont distincts et qui ont pourtant le même comportement applicatif. La relation $=_{\beta\eta}$ que nous allons introduire dans cette section permet de gommer ce phénomène : tout terme "sera" une abstraction (à $\beta\eta$-équivalence près) et deux éléments de $\Lambda/=_{\beta\eta}$ ayant le même comportement applicatif seront nécessairement égaux (on appelle cette propriété "l'extensionnalité").

Définition 70 *On appelle* η-redex *tout terme de L de la forme $\lambda x.(t)x$, où $x \notin Vlib(t)$. Son* réduit *est t, par définition.*

On remarquera que si $t \equiv \lambda x.u$ alors $\lambda x.(t)x$ se β_0-réduit à $\lambda x.u \equiv t$. Donc cette notion n'apporte rien de nouveau pour les termes qui sont des abstractions.

Lemme 71 *Tout terme α-équivalent à un η-redex est un η-redex et leurs deux réduits sont α-équivalents (la notion passe donc à Λ).*

Définition 72 *Un $\beta\eta$-redex est un β-redex ou un η-redex.*

η_0 ou $\to_{\eta_0}$ est la relation binaire définie sur Λ par : $t\eta_0 t'$ si t' est obtenu en contractant un η-redex de t.

η ou $\to_\eta$ est la clôture réflexive et transitive de η_0.

$\beta\eta$ ou $\to_{\beta\eta}$ se définit de façon analogue à partir de $\beta_0 \cup \eta_0$ (ou $\beta \cup \eta$).

Les notions de : terme *η-normal*, *$\beta\eta$-normal* (resp. *normalisable*), *η-équivalence* ($=_\eta$), *$\beta\eta$-équivalence* ($=_{\beta\eta}$), etc. se définissent de la même manière que pour β.

Remarque 73 *$u \to_{\eta_0} u'$ implique $l(u') < l(u)$; il en résulte que tout terme est η-normalisable.*

Exemple 74 *1. Tout entier de Church $\hat{n}$ tel que $n \neq 1$ est $\beta\eta$-normal*
2. L'entier de Church $\hat{1}$ se η_0-réduit à I, qui est $\beta\eta$-normal.
3. $I =_{def} \lambda x.x$ et $\varepsilon =_{def} \lambda x.\lambda y.xy$ sont deux termes β-normaux η-equivalents.

Dans ce cours nous travaillerons principalement sur β, mais les faits suivants sont utiles à connaître et faciles à retenir. Ils se démontrent de façon analogue au cas β (Pour Church-Rosser voir la section suivante).

Proposition 75 *La η- et la $\beta\eta$- réductions sont contextuelles et substitutives.*

Théorème 76 *(Church-Rosser) La η et la $\beta\eta$- réductions sont confluentes.*

Corollaire 77 *Deux termes sont $\beta\eta$-équivalents ssi ils ont un $\beta\eta$-réduit commun (vrai aussi pour η).*

Corollaire 78 *$\Lambda / =_{\beta\eta}$ et $\Lambda^\circ / =_{\beta\eta}$ sont infinis.*

Preuve..Il suit facilement de Church-Rosser et des propriétés triviales des entiers de Church que nous avons mentionnées ci-dessus, que deux entiers de Church distincts ne peuvent pas être $\beta\eta$-équivalents.

Proposition 79 *La $\beta\eta$-équivalence est contextuelle et substitutive.*

Comme pour la β-équivalence la Proposition 75 nous dit en particulier que les opérations d'application et d'abstraction, ainsi que de substitution, sont compatibles avec la $\beta\eta$-équivalence et passent au quotient. On obtient ainsi un nouveau modèle syntaxique du λ-calcul, basé cette fois sur $\Lambda / =_{\beta\eta}$. On l'appelle *modèle des termes du λ-calcul extensionnel,* où le mot “extensionnel” réfère à la propriété suivante.

Proposition 80 *(extensionnalité)*
$\forall u\,(tu =_{\beta\eta} t'u) \Longrightarrow t =_{\beta\eta} t'$

Preuve. Supposons en effet que $tx =_{\beta\eta} t'x$ avec $x \notin Vlib(t, t')$, alors $\lambda x.tx =_{\beta\eta} \lambda x.t'x$, et donc $t =_{\beta\eta} t'$.

Le premier modèle de Scott, noté $\mathcal{D}^\infty$, était en fait un modèle du λ-calcul extensionnel, en ce sens qu'il satisfaisait la propriété du premier ordre correspondante : $\mathcal{D}^\infty \models \forall d \forall d' \,[(\forall e(de = d'e) \Rightarrow (d = d')]$. [6]

Lemme 81 *u normal (i.e. β-normal) et $u \to_\eta u'$ impliquent u' β-normal.*

Preuve. Induction sur la longueur (structure) du terme normal. **A faire.**

Proposition 82 *Un terme u est $\beta\eta$-normalisable ssi il est β-normalisable.*

Preuve. $\Leftarrow$. Soit n β-normal tel que $u \to_\beta n$; comme tout terme est η-normalisable il existe n' η-normal tel que $n \to_\eta n'$; d'après le lemme précédent précédent n' est $\beta\eta$-normal.

$\Rightarrow$. Une démonstration syntaxique est donnée dans [Bar, ch.15 §1], mais nous donnerons une démonstration alternative de ce résultat comme corollaire dans un chapitre suivant.

[6] Noter que pour exprimer cette propriété il suffit de disposer d'un symbole de fonction binaire "." et qu'on a abrégé ici $d.e$ en de.

4 La preuve du Théorème de Church-Rosser.

Pour prouver le Théorème de Church-Rosser, qui affirme que $\rightarrow_\beta$ est confluente, il suffit, compte tenu du Lemme 54, de trouver une relation confluente dont $\rightarrow_\beta$ est la clôture transitive, sachant que β_0 ne peut pas faire l'affaire puisque β_0 n'est pas confluente (regarder les deux β_0-réduits évidents de δR, où R est un β-redex). La solution c'est de remplacer β_0 par la relation suivante, notée $\Rightarrow_0$, dont l'idée intuitive c'est que l'on se donne le droit de réduire simultanément autant des redex présents dans le terme que l'on veut (réduction "en parallèle"), ou de ne pas en réduire du tout (réflexivité). Cette notion de réduction parallèle a été introduite par Tait et Martin-Lof (dans le but de prouver Church-Rosser). On procèdera de la même manière avec η et $\beta\eta$.

Définition 83 *On définit l'ensemble $B(t)$ des réduits de t pour $\Rightarrow_0$, par induction sur $l(t)$:*
1. si $t \in V$ alors $B(t) =_{def} \{t\}$
2. si $t = \lambda x.u$ alors $t' \in B(t)$ si et seulement si $t' = \lambda x.u'$ avec $u' \in B(u)$
3. si $t = (u)v$ alors $t' \in B(t)$ ssi on est dans un des 2 cas suivants :
a) $t' = (u')v'$ avec $u' \in B(u)$, $v' \in B(v)$.
b) $u = \lambda x.w$ et $t' = w'[x := v']$ avec $w' \in B(w)$, $v' \in B(v)$

Remarque 84 *a) $\Rightarrow_0$ est contextuelle (et en particulier réflexive),*
b) $\Rightarrow_0$ contient les couples (R, S) où R est un β-redex et S son réduit,
c) $\Rightarrow_0$ n'est pas transitive.

La non transitivité vient de ce que $\Rightarrow_0$ ne réduit que les redex présents, pas ceux qu'elle créera éventuellement. Pour un exemple concret regarder par exemple ce qui se passe avec $t \equiv \delta I$.

Lemme 85 *1. $t \rightarrow_{\beta 0} t'$ implique $t \Rightarrow_0 t'$*
2. $t \Rightarrow_0 t'$ implique $t \rightarrow_\beta t'$

Preuve. 1. Cela revient à montrer que, pour tout t, on a $A(t) \subseteq B(t)$ (cf. Définition 46) ; induction facile.

2. se démontre par induction sur t, en utilisant le fait que β est contextuelle et, dans le cas où t est un redex, qu'elle est substitutive.

Corollaire 86 *$\rightarrow_\beta$ est la clôture transitive de $\Rightarrow_0$.*

Preuve. En effet $\beta_0 \subseteq \Rightarrow_0 \subseteq \beta$, et β est la clôture réflexive et transitive de β_0.

Plus en détail : $\beta =_{def} \beta_0^{rt} \subseteq \Rightarrow_0^{rt} = \Rightarrow_0^{t} \subseteq \beta$. La première inclusion suit de la croissance de l'opérateur $(-)^{rt}$ sur $P(\Lambda \times \Lambda)$, l'égalité $\Rightarrow_0^{rt} = \Rightarrow_0^{t}$ suit du fait que $\Rightarrow_0$ est déja réflexive, et enfin la dernière inclusion vient de ce que β est transitive et contient $\Rightarrow_0$, donc elle contient sa clôture transitive.

Lemme 87 *$t \Rightarrow_0 t'$ et $\bar{v} \Rightarrow_0 \bar{v}'$ impliquent $t[\bar{x} := \bar{v}] \Rightarrow_0 t'[\bar{x} := \bar{v}']$.*
en particulier $\Rightarrow_0$ est substitutive.

Preuve. **A Faire en cours ou à faire rédiger.**
Par induction sur t en utilisant le Lemme 39 dans le cas où t est un redex.

Proposition 88 *Il existe une fonction $t \mapsto t^*$ sur Λ qui est telle que*
$\forall u\,(t \Rightarrow_0 u$ *implique* $u \Rightarrow_0 t^*)$

Preuve. On définit t^* par induction sur t de la manière suivante :
$(x)^* =_{def} x$
$(\lambda x.t)^* =_{def} \lambda x.t^*$
$(vw)^* =_{def} v^*w^*$ si vw n'est pas un redex
$(vw)^* =_{def} r^*[x := w^*]$ si $v \equiv \lambda x.r$.
Intuitivement, t^* est obtenu à partir de t en réduisant simultanément tous les β-redex présents dans t (on dit que t^* est un "*développement complet*" de t).
Il reste à prouver que cette fonction satisfait bien la conclusion de la proposition.
On raisonne une fois de plus par induction sur $l(t)$.
Le seul cas non trivial est le cas où $t \equiv (\lambda x.r)w$ et dans ce cas $t^* =_{def} r^*[x := w^*]$. Supposons que $t \Rightarrow_0 u$; alors $u = (\lambda x.r')w'$ ou $u = r'[x := w']$, avec $r \Rightarrow_0 r'$ et $w \Rightarrow_0 w'$. Par hypothèse d'induction on sait que $r' \Rightarrow_0 r^*$ et $w' \Rightarrow_0 w^*$. Dans les deux cas on a $u \Rightarrow_0 r^*[x := w^*] =_{def} t^*$: dans le premier cas cela suit de la définition même de $\Rightarrow_0$, et, dans le second, du Lemme 87.

Corollaire 89 $\Rightarrow_0$ *est confluente.*

Remarque 90 $\omega^* = \omega$.

Modifications à apporter dans le preuve de Church-Rosser pour $\beta\eta$. On définit une réduction parallèle $\Rightarrow_0^\eta$. L'ensemble $B(t)$ des $\Rightarrow_0^\eta$-réduits de t est défini en remplaçant la clause 2. de la définition de $B(t)$ (Définition 83) par :
2^η. si $t = \lambda x.u$ alors $t' \in B(t)$ si et seulement si on est dans un des deux cas suivants :
a) $t' = \lambda x.u'$ avec $u' \in B(u)$, ou
b) $t' = v'$ si $u = vx$ avec $x \notin Vlib(v)$, et si $v' \in B(v)$.
On modifie de plus la définition de la fonction "*" sur les abstractions, de la façon suivante :
a) si t est de la forme vx avec $x \notin Vlib(v)$ alors $(\lambda x.t)^* =_{def} v^*$
b) sinon $(\lambda x.t)^* =_{def} \lambda x.t^*$.

5 λ-congruences.

Définition 91 *Une* congruence sur Λ *est une relation d'équivalence contextuelle sur Λ. La congruence $\Lambda \times \Lambda$ est dite* triviale. *Une* λ-congruence *est une congruence qui contient la β-équivalence. Une* $\lambda\eta$-congruence *est une congruence qui contient la $\beta\eta$-équivalence. Dans les deux cas il suffit en fait de demander que la congruence contienne les couples (R, S) où R est un redex et S est son réduit.*

Exercice 92 *Une λ-congruence $\sim$ est triviale ssi $V \sim F$ ssi $x \sim y$.*

Exercice 93 *Une λ-congruence $\sim$ est une $\lambda\eta$-congruence ssi $\varepsilon \sim I$ (rappel : $\varepsilon =_{def} \lambda x.\lambda y.xy$).*

Les sources de ce chapitre sont les suivantes :

Références

[1] H. Barendregt, The lambda-calculus, its syntax and semantics, Studies in Logic 103, North Holland, revised ed. 1984.

[2] J.L. Krivine, Lambda-calcul, types et modèles, Masson 1990, *pour le traitement de la α-équivalence et des substitutions* (il existe une version augmentée en anglais : Lambda-calculus, types and models, Ellis et Horwood 93)

[3] S. Ronchi della Rocca, Basic lambda-calculus (notes de cours, Université de Turin, prépublication), *pour la formulation de la preuve de Church-Rosser,* elle-même basée sur la présentation de M. Takahshi, “parallel reduction in λ-calculus, Tokyo Institute of Technology (internal report), et pour la note historique correspondante.