Le lambda calcul

Notes de cours

A l'attention des étudiants de L3 Informatique,

Université de Bourgogne, année 2013-2014.

Par O. Bailleux

Fonctions récursives

Rappel sur le fonctionnement de la machine de Turing.

Une MT à n états est constituée des éléments suivants :

Une bande supposée infinie constituée de cases contenant des 0 et des 1

Une tête mobile de lecture / écriture

Une variable d'état pouvant contenir une valeur parmi 1..n-1, start, stop.

Un programme...

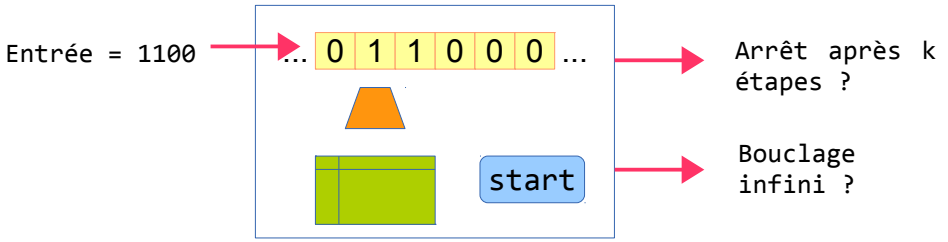
Programme d'une MT à n états

Il est constitué de 2n lignes, i.e., 2 lignes pour chaque valeur possible de la variable d'état (incluant la valeur start, mais pas stop).
Chaque ligne dit ce qu'il faut faire pour un état donné et une valeur donnée lue sur la bande par la tête : valeur à écrire sur la bande (0 ou 1), déplacement de la tête (gauche ou droite), et nouvel état.

Les MT qui s'arrêtent et les autres

Il y a 4(n+1)^(2n) MT à n états. Pour chacune, il y a une infinité de valeurs initiales possibles de la bande.

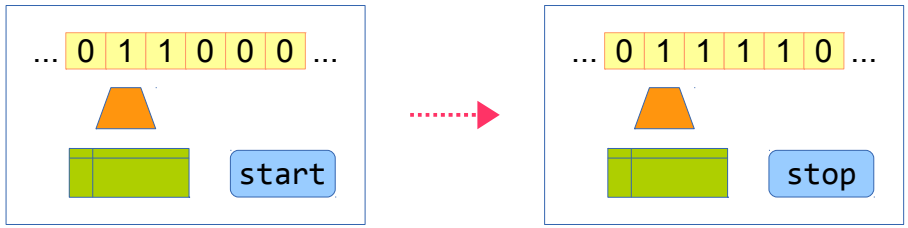
Pour chaque valeur initiale de la bande, une MT peut soit s'arrêter après un nombre fini d'étapes de calcul, soit boucler indéfiniment.



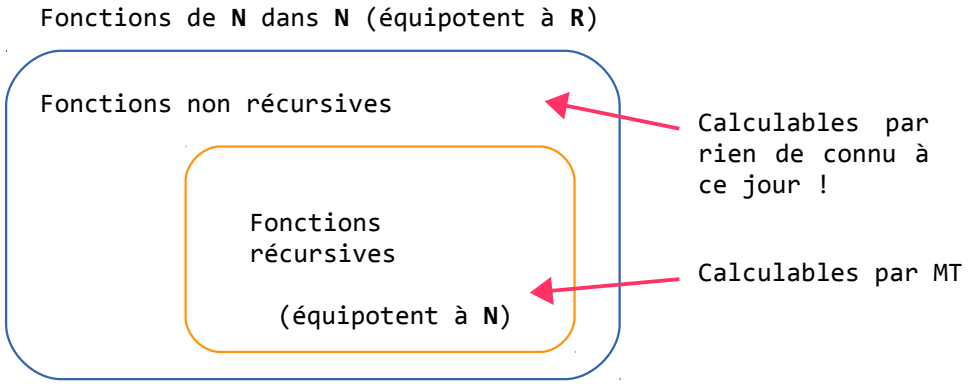
Les MT qui calculent des fonctions

Certaines MT calculent des fonctions : la valeur d'entrée (codée de manière appropriée) est placée sur la bande, la valeur de sortie est récupérée sur la bande après arrêt.
Une telle MT doit s'arrêter pour toute valeur d'entrée appartenant à l'ensemble de départ de la fonction à calculer.

$$f : N \rightarrow N \, , \, f(x) = x^2$$



Toutes les fonctions ne sont pas calculables par MT



Une fonction non récursive : S

A tout entier n, la fonction S associe le plus grand nombre d'étapes que peut effectuer une MT à n état sans boucler indéfiniment et avec une bande initialement remplie de 0.

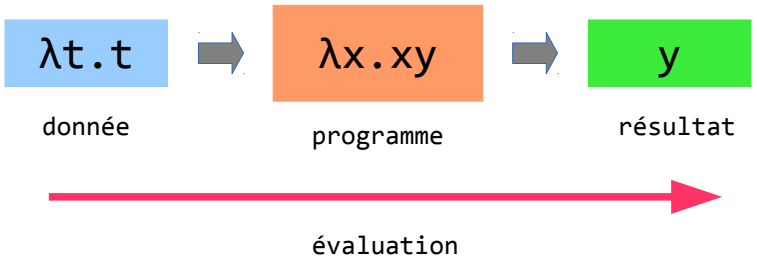
n	S(n)
1	1
2	6
3	21
4	107
5	>= 47 176 870
6	> 1,29 10^1730

S a une particularité remarquable :

elle croît plus vite que n'importe quelle fonction calculable !

Le lambda calcul

C'est un langage permettant de réaliser des spécifications exécutables de fonctions, c'est à dire des programmes.

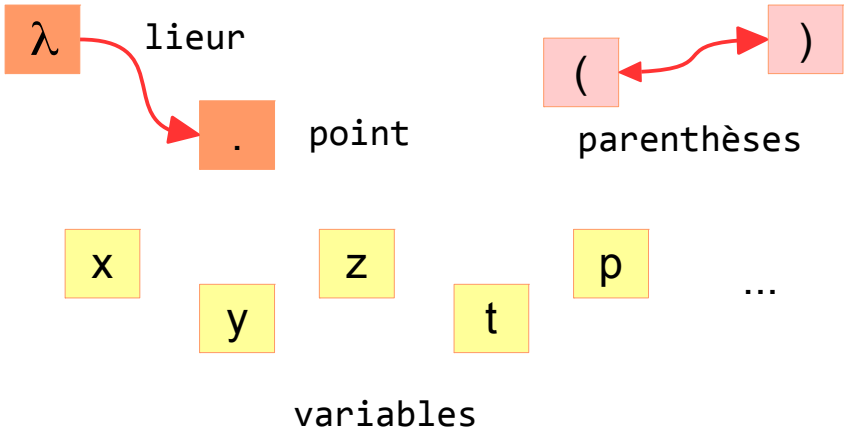


Un système de calcul universel

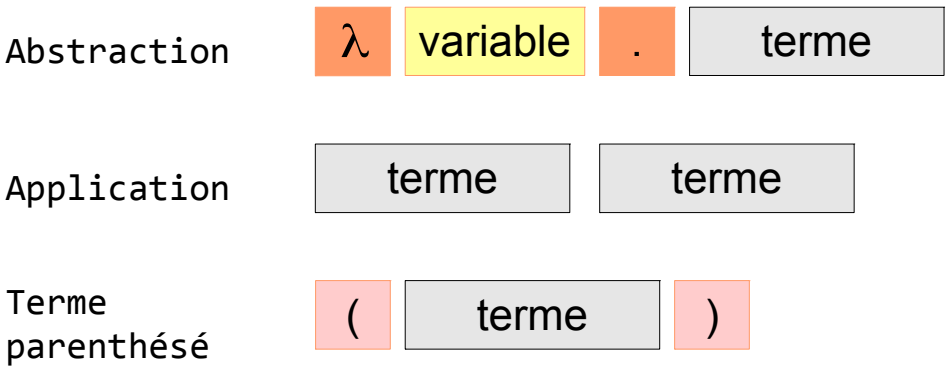
Le lambda-calcul permet de réaliser des spécifications exécutables permettant de calculer toute fonction récursive (en adoptant une représentation appropriée des informations).

Il fait partie des **systèmes de calcul universels**, comme la MT, le jeu de la vie, le langage C, Java... et est l'ancêtre des langages fonctionnels.

Les briques du lambda-calcul



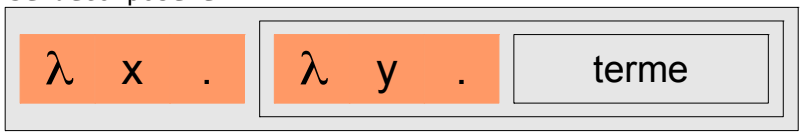
Les termes du lambda-calcul



Décomposition des termes

Associativité à droite des abstractions

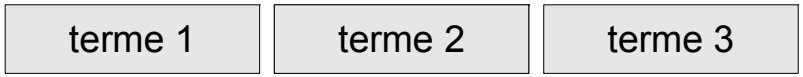
Se décompose en



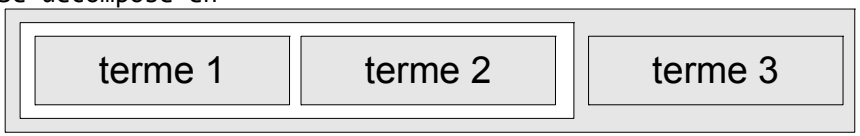
Est équivalent à

Décomposition des termes

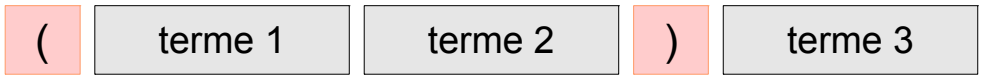
Associativité à gauche des applications



Se décompose en

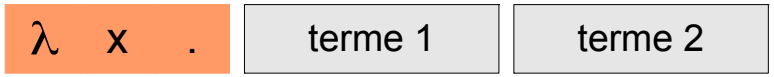


Est équivalent à

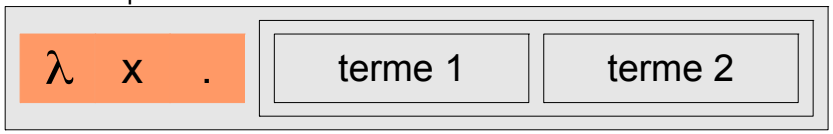


Décomposition des termes

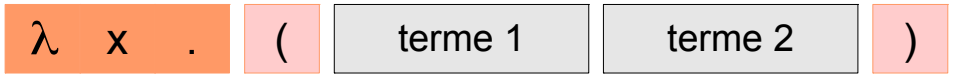
Priorité de l'application par rapport à l'abstraction



Se décompose en

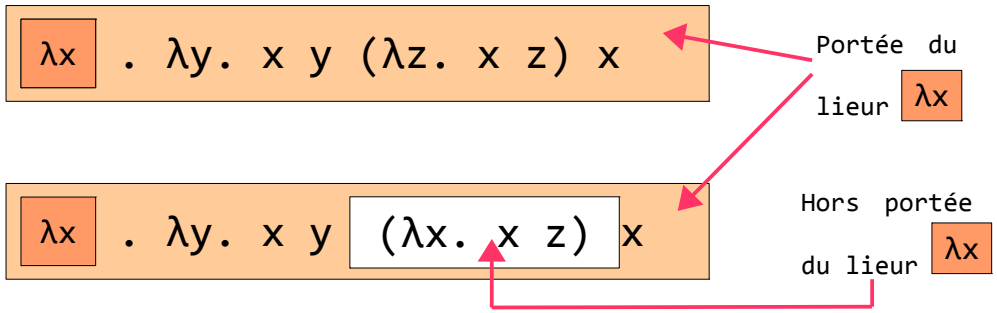


Est équivalent à



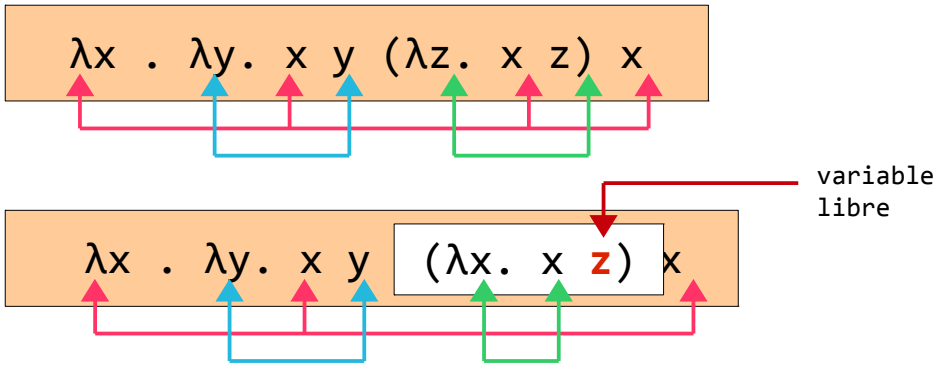
Portée d'un lieur

La portée d'un lieur associé à une variable de liaison est l'abstraction introduite par ce lieur, à l'exception des abstractions incluses introduites avec la *même* variable de liaison.



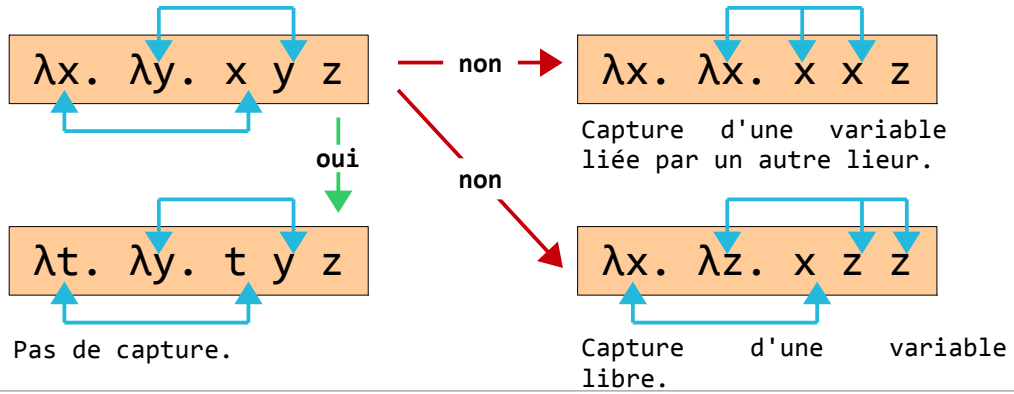
Variables libres et liées

Toutes les occurrences d'une variable de liaison situées dans la portée du lieur associé sont dites liées entre elles. Les variables non liées sont dites libres.



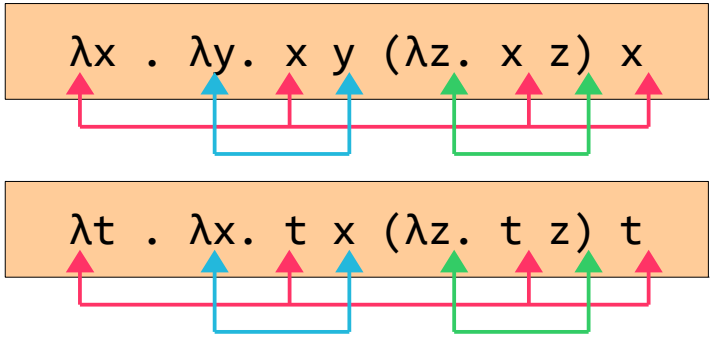
Renommage

Un renommage de toutes les occurrences des variables liées par un lieur est autorisé s'il ne provoque pas de capture de variables libres ou de variables liées par un autre lieur.



Termes équivalents

Deux termes sont dits équivalents si et seulement si chacun des deux peut être obtenu à partir de l'autre par un renommage autorisé et/ou une modification du parenthésage préservant la structure.



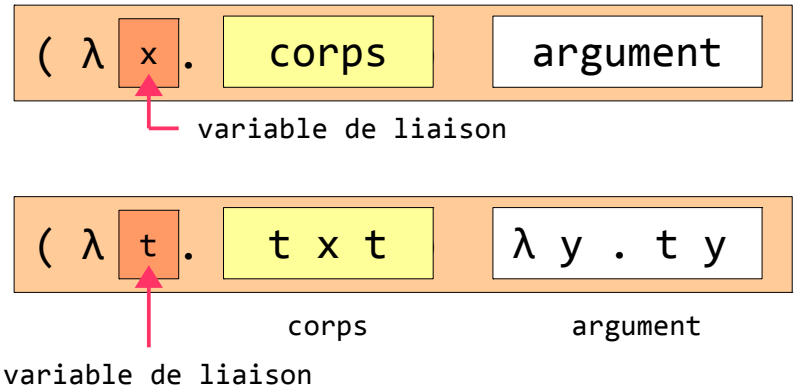
termes équivalents

Entraînement

λx.λy.xy(λz.xz)x	?	λx.λy.xy(λx.xx)x
λx.λy.xy(λz.xz)x	?	λx.λy.xy λz.xz x
λx.λy.xy(λz.xz)x	?	λx.λy.(xy)(λz.xz)x
λx.λy.xy(λz.xz)x	?	λx.λz.xz(λz.xz)x

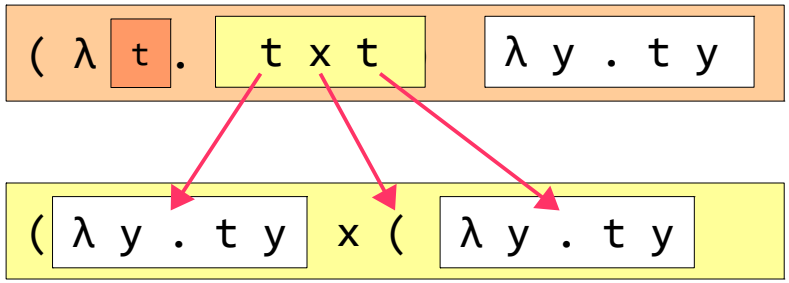
Redex

Un redex est une abstraction entre parenthèses suivie d'un terme.



Réduction

Une réduction consiste à remplacer un redex par son corps dans lequel chaque occurrence de la variable de liaison est remplacée par l'argument mis entre parenthèses.



Entraînement

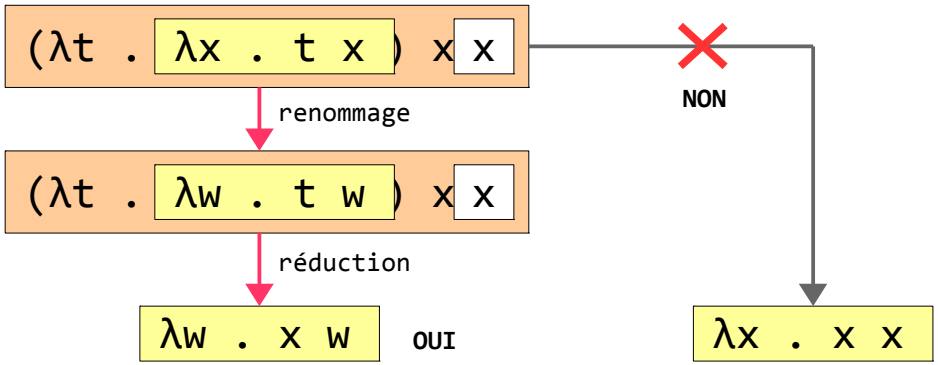
Peut on réduire le terme ci dessous et si oui quel est le résultat ?

$(\lambda y\ .\ t\ y)\ x\ (\lambda y\ .\ t\ y)$

1. Aucune réduction n'est possible.
2. $t\ x\ (\lambda y\ .\ t\ y)$
3. $t\ (\ x\ (\lambda y\ .\ t\ y)\ )$

Cas particulier exigeant un renommage

Un renommage est parfois nécessaire pour éviter qu'une variable libre de l'argument soit capturé lors de la réduction.



Entraînement

Pour chacun des termes ci dessous, dites si un renommage est nécessaire (réponse 1) ou pas (réponse 2).

a `(λx . x x) (λy . x y)`

b `(λx . λy . x y) x λx . x`

c `(λx . λy . x y) λt . y t`

Évaluation d'un terme

Évaluer un terme consiste à réaliser des réductions tant que c'est possible. Le résultat ne dépend pas de l'ordre (si applicable) dans lequel les réductions sont faites.

Exemple :

`(λx . λy . y x) a b`

A compléter...