Implémentation de la β-réduction dans une machine de Turing

Antoine Dequay

21 septembre 2022

Notes

— Prof : Lilian Besson.

— Leçon : 913, 929.

Théorème 1 On peut implémenter la β-réduction dans une machine de TURING.

Plus précisément, on peut construire une machine de Turing à 7 rubans de travail semi infinis (avec des têtes de lectures indépendantes) qui, étant donné un λ-terme en entrée, détecte s'il est sous forme normale, et réduit le redex le plus à gauche dans le cas contraire.

Preuve. **Définition 2** *Indices de* DE BRUIJN. On peut représenter les λ-termes via les indices de DE BRUIJN : Pour une formule donnée, notons $\{l_i, i \in \mathbb{N}^*\}$ les variables libres, et $\{v_i, i \in \mathbb{N}^*\}$ variables liées, où l'indice de v_i indique à quel λ cette variable est liée (en les numérotant de droite à gauche depuis v_i). C'est donc le nombre de λ présents entre la variable liée et son λ correspondant.

Par exemple, $\lambda ab.bac$ se traduit par $\lambda\lambda v_1 v_2 l_1$.

Pour représenter les λ-termes sur un alphabet fini, on va considérer un codage unaire des entiers, ce qui donne l'ensemble de variables $\mathcal{V} = (v|l)1^+$. On utilise une notation préfixe pour s'affranchir de l'ambiguïté et des parenthèses. En mettant de côté les considérations sur les variables libres ou liées, les λ-termes seront alors des mots de $\Lambda := \mathcal{V}|@\Lambda\Lambda|\lambda\Lambda$.

Notre exemple s'écrit alors $\lambda\lambda@@v1v11l1$.

On travaille donc sur l'alphabet $\Sigma = \{v, l, 1, \lambda, @\}$ dans la suite.

Commençons par remarquer qu'un redex du type $(\lambda x.t_1)t_2$ se traduit par la présence de $@\lambda$ au début de son écriture.

Donnons-nous un λ-terme écrit via les transformations ci-dessus sur la première première bande. On appelle les six autres bandes dans l'ordre : pré-redex, fonction, argument, post-redex, réduction et travail.

Décrivons un cycle d'exécution :

1. La machine lit la première bande et repère ou non un redex de la forme $@\lambda$. S'il n'y en a pas, la machine s'arrête. Sinon :
   - La partie fonctionnelle du redex est recopiée dans la bande "fonction",
   - L'argument est recopié dans la bande "argument",
   - Ce qui apparaît avant (resp. après) le redex est recopié sur la bande "pré-redex" (resp. "post-redex").
2. La machine copie le contenu de "fonction" dans "réduction" en omettant le λ initial et en remplaçant les occurrences de la variable liée par le contenu de "argument",
3. Le contenu de la première bande est remplacé par la concaténation de "pré-redex" auquel on a enlevé son @ final, "réduction" et "post-redex",
4. On efface le contenu des autres lignes.

Il y a un point important à voir : **Comment isoler "fonction" et "argument"**.

On cherche à isoler les chaînes associées à t_1 et t_2 dans le redex $(\lambda x.t_1)t_2$. Pour ce faire, il suffit de comparer le nombre de @ et de variables rencontrées en lisant le ruban d'entrée :

En effet, si on regarde l'arbre syntaxique d'un λ-terme, chaque @ induit deux sous-arbres, et on peut alors montrer par induction que le nombre de feuilles (ici le nombre de variables), est égal au nombre de nœuds binaires (ici les @) plus un.

En lisant l'encodage préfixe de cet arbre, cette propriété n'est vraie que sur l'expression complète, et tout préfixe de l'expression a au moins autant de @ que de variables (c'est analogue au nombre de parenthèses ouvrantes et fermantes dans un bon parenthèsage). Elle est en particulier vraie au sein des sous-arbres associés aux termes $\lambda x.t_1$ et t_2.

Ainsi, on définit un compteur dans "travail", initialisé à 0, et on commence la recopie du sous-terme dans "fonction". A chaque fois que l'on rencontre un @, on incrémente le compteur de "travail", et à chaque fois qu'on rencontre un v ou un l on le décrémente. On arrête la recopie quand on cherche à décrémenter 0. Dans ce cas, on passe à la recopie dans "argument" via la même méthode, puis à la recopie dans "post-redex".

Un dernier point est de savoir **quelles sont les variables à remplacer lors de la réduction**.

Lorsqu'on parcours la fonction lors de la recopie, on peut utiliser le ruban de travail en écrivant un g à chaque @ rencontré et un 1 à chaque λ rencontré. Vérifier que la variable rencontrée est liée au premier λ revient à comparer le nombre de 1 dans le ruban de travail à l'indice de la variable.

A chaque nouvelle variable rencontrée, on efface le ruban de travail de droite à gauche jusqu'à rencontrer un g (on était dans le sous-arbre gauche d'un @), qu'on remplace par un d avant de continuer.

□

Exemple 3 Faisons "tourner" notre machine sur $\lambda a.((\lambda cd.(aed))\ (\lambda g.g))\ \ (\lambda ab.a)$.