Le théorème d'accélération de Blum

Paul MERCAT

11 février 2007

Introduction

Ce document présente une étude de la complexité, en partant des axiomes proposés par Blum, pour définir la complexité. Après avoir donné la définition et des exemples, je donnerai quelques propriétés, comme l'existence de fonctions aussi complexes que l'on veut, des liens qu'il peut y avoir entre complexité et valeurs, des liens qui existe entre deux mesures de complexité, et le théorème de combinaison. Je présenterai ensuite le thérorème d'accélération de Blum, qui donne l'existence de fonctions que l'on peut sans arrêt accélerer. Il existe donc des fonctions sans meilleur programme.

1 Mesure de compléxité

Définition : On appelle mesure de compléxité un couple $((\varphi_i)_{i\in\mathbb{N}}, (c_i)_{i\in\mathbb{N}})$, où $(\varphi_i)_{i\in\mathbb{N}}$ est une énumération des fonctions récursives de $\mathbb{N}$ dans $\mathbb{N}$, et $(c_i)_{i\in\mathbb{N}}$ est une suite de fonctions récursives de $\mathbb{N}$ dans $\mathbb{N}$ vérifiant les deux hypothèses :

1. $c_i(n)$ est définie si et seulement si $\varphi_i(n)$ est définie
2. La fonction $M : \mathbb{N}^3 \to \mathbb{N}$ définie par :

$$M(i,n,m) = \begin{cases} 1 & \text{si} \quad c_i(n) = m \\ 0 & \text{sinon} \end{cases}$$

est récursive totale ($M(i,n,m) = 0$ si $c_i(n)$ n'est pas défini).

Une fonction c_i correspond à la complexité de la fonction φ_i.
On a vu en Logique, que l'on pouvait trouver une énumération des fonctions récursives. Pour obtenir une mesure de complexité, il suffit donc de trouver des compléxités $(c_i)_{i\in\mathbb{N}}$ qui vérifies bien les axiomes.

Exemples :

1. Le nombre d' étapes de calculs d'une machine de Turing donne une mesure de complexité (Il est facile de transformer une fonction récursive en machine de Turing, de façon récursive).
2. En posant $\forall i, n \in \mathbb{N}, c_i(n) = 0$, le premier point n'est pas vérifié.
3. En posant $\forall i, n \in \mathbb{N}, c_i(n) = 0$ si $\varphi_i(n)$ est défini et $c_i(n)$ non défini sinon, le deuxième point n'est pas vérifié (sinon, le problème de terminaison serait décidable).
4. Etant donné un ordinateur, le nombre d'instructions assembleur éxécutées fourni une mesure de complexité (Il est facile d'obtenir un programme à partir d'une fonction récursive, de manière récursive).

Notation : Je dirais qu' un entier i est un indice pour la fonction récursive ψ, si $\forall n \in \mathbb{N}, \psi(n) = \varphi_i(n)$.

Je donne maintenant quelques résultats sur les mesures de complexité, afin de manipuler un peu cette nouvelle notion. Le théorème qui suit donne l'existence de fonctions aussi complexes que l'on veut, pour toute mesure de complexité.

Théorème 1 : Soit $((\varphi_i)_{i\in\mathbb{N}}, (c_i)_{i\in\mathbb{N}})$ une mesure de complexité. Soit $f : \mathbb{N} \to \mathbb{N}$ une fonction récursive totale. Alors, il existe une fonction récursive totale $\psi : \mathbb{N} \to \{0, 1\}$ telle que pour tout indice i pour ψ, on ai $c_i(n) > f(n)$ pour presque tout n.

Preuve Soit $u : \mathbb{N} \to \mathbb{N}$ la fonction récursive définie par :

$$u(n) = \begin{cases} 1 & \text{si} \quad n = 0 \\ 0 & \text{sinon} \end{cases}$$

Construisons la fonction ψ de la façon suivante :

$$\psi(0) = \begin{cases} u(\varphi_0(0)) & \text{si} \quad c_0(0) \leq f(0) \rightarrow \quad \text{marquer 0} \\ 0 & \text{sinon} \end{cases}$$

$$\psi(n+1) = \begin{cases} u(\varphi_i(n)) & \text{où } i \text{ est le plus petit indice} \leq n \text{ non déja marqué} \\ & \quad \text{tel que } c_i(n) \leq f(n) \rightarrow \text{ marquer } i \\ 0 & \text{sinon} \end{cases}$$

La fonction ψ est récursive, et si i est un indice pour ψ, alors il existe un rang n_0 à partir duquel tous les indices inférieurs à i qui auraient été marqués sont marqués. On a alors, $\forall n > n_0, c_i(n) > f(n)$, puisque i n'est jamais marqué. □

Voyons maintenant les liens qu'il peut y avoir entre les valeurs d'une fonction et sa complexité.

Théorème 2 : Soit $((\varphi_i)_{i\in\mathbb{N}}, (c_i)_{i\in\mathbb{N}})$ une mesure de complexité.

1. Il n'existe pas de fonction récursive $k : \mathbb{N}^2 \to \mathbb{N}$ telle que $\forall i \in \mathbb{N}, \forall n \in \mathbb{N}, k(n, \varphi_i(n)) \geq c_i(n)$
2. Il existe une fonction récursive $h : \mathbb{N}^2 \to \mathbb{N}$ telle que $\forall i \in \mathbb{N}, h(n, c_i(n)) \geq \varphi_i(n)$ pour presque tout n.

Preuve Le premier point est une conséquence du théorème précédent. On sait que la fonction H définie par :

$$H(i, n, m) = \begin{cases} \varphi_i(n) & \text{si} \quad c_i(n) = m \\ 0 & \text{sinon} \end{cases}$$

est récursive totale. La fonction $h(n, m) = \max_{i\leq n} H(i, n, m)$ est donc récursive totale. Et on a $h(n, c_i(n)) \geq \varphi_i(n)$ dès que $n \geq i$. □

Le théorème qui suit affirme que deux mesures de complexité sont récursivement liées. Il sera utilisé dans la preuve du théorème d'accélération de Blum.

Théorème 3 : Soient $((\varphi_i)_{i\in\mathbb{N}}, (c_i)_{i\in\mathbb{N}})$ et $((\varphi_i)_{i\in\mathbb{N}}, (c'_i)_{i\in\mathbb{N}})$ deux mesures de complexité. Alors, il existe une fonction récursive totale $r : \mathbb{N}^2 \to \mathbb{N}$ telle que $\forall i \in \mathbb{N}, r(n, c_i(n)) \geq c'_i(n)$ et $r(n, c'_i(n)) \geq c_i(n)$ pour presque tout n.

Preuve Soit $H : \mathbb{N}^3 \to \mathbb{N}$ définie par :

$$H(i,n,m) = \begin{cases} \max(c_i(n), c'_i(n)) & \text{si } c_i(n) = m \text{ ou } c'_i(n) = m \\ 0 & \text{sinon} \end{cases}$$

Posons $r(n,m) = \max_{i\leq n} H(i,n,m)$. La fonction r est récursive totale ($c_i(n)$ définie $\Leftrightarrow$ $\varphi_i(n)$ définie $\Leftrightarrow$ $c'_i(n)$ définie), et on a $r(n, c_i(n)) \geq c'_i(n)$ et $r(n, c'_i(n)) \geq c_i(n)$ dès que $n \geq i$. $\square$

Le théorème qui suit donne une propriété supplémentaire des mesures de complexité.

Théorème 4 (de combinaison) : Soit $((\varphi_i)_{i\in\mathbb{N}}, (c_i)_{i\in\mathbb{N}})$ une mesure de complexité. Soit $c : \mathbb{N}^2 \to \mathbb{N}$ une fonction récursive telle que : $\varphi_i(n)$ et $\varphi_j(n)$ sont définies $\Rightarrow \varphi_{c(i,j)}(n)$ est définie. Alors, il existe une fonction récursive totale $r : \mathbb{N}^3 \to \mathbb{N}$ telle que : $\forall i \in \mathbb{N}, r(n, c_i(n), c_j(n)) \geq c_{c(i,j)}(n)$ pour presque tout n.

Preuve La fonction $H : \mathbb{N}^5 \to \mathbb{N}$ définie par :

$$H(i,j,n,m1,m2) = \begin{cases} c_{c(i,j)} & \text{si } \quad c_i(n) = m1 \text{ et } c_j(n) = m2 \\ \text{non définie} & \text{sinon} \end{cases}$$

est récursive totale. Soit $r(n, m1, m2) = \max_{i,j\leq n} H(i,j,n,m1,m2)$. Alors $r(n, c_i(n), c_j(n)) \geq c_{c(i,j)}(n)$ dès que $n \geq \max(i,j)$. $\square$

2 Le théorème d'accéleration de Blum

Dans la partie précédente, nous avons vu quelques propriétés des mesures de complexité. Nous allons maintenant voir qu'il existe des fonctions que l'on peut accélerer d'autant que l'on veut, autant de fois que l'on veut. J'établirai d'abord le résultat pour une mesure de complexité particulière, puis je généraliserai le résultat à toute mesure de complexité, grâce au théorème 3.

Notation : Je noterai $((\varphi_i)_{i\in\mathbb{N}}, (l_i)_{i\in\mathbb{N}})$ la mesure de complexité obtenue en prenant $(\varphi_i)_{i\in\mathbb{N}}$ une énumération quelconque des fonctions récursives, et en prenant $(l_i)_{i\in\mathbb{N}}$ telle que $l_i(n)$ est le nombre de cases utilisées par la machine de Turing à une bande obtenue récursivement à partir de la fonction φ_i, telle qu'elle ne boucle sur aucune partie finie de la bande.

Le résultat suivant donne l'existence d'une récursive aussi grande que l'on veut, dont la complexité est majorée par sa valeur. Ce résultat servira dans la démonstration du deuxième lemme.

Lemme : Pour la mesure de complexité $((\varphi_i)_{i\in\mathbb{N}}, (l_i)_{i\in\mathbb{N}})$, et pour toute fonction récursive totale $r : \mathbb{N} \to \mathbb{N}$, il existe une fonction récursive totale $f : \mathbb{N} \to \mathbb{N}$ telle que si c est sa complexité, et cr celle de la fonction r, on ait $c \leq 2 \cdot f$, $r \leq f$, $cr \leq f$, et $f(n) \geq n^2$.

Preuve Si r ne vérifie pas $\forall n \in \mathbb{N}, r(n) \geq n^2$, il suffit de la remplacer par : $r'(n) = n^2 + \max_{i \leq n} r(i)^2$. On définit la fonction f par : $f(n) = \max(r(n), c(n))$. Montrons que l'on peut calculer $f(n)$ en utilisant un espace inférieur ou égal à $r(n) + c(n)$. Voici les étapes que l'on suit pour ainsi calculer $f(n)$:

1. on calcule $r(n)$ (ce qui utilise $c(n)$ cases du ruban)
2. on calcule le nombre de cases qui ont été utilisées, en écrivant le résultat à la suite (ce qui utilise au plus $r(n)$ cases)
3. on compare les deux nombres et on rend le plus grand (ce qui n'utilise pas d'espace supplémentaire)

La compléxité du calcul de $f(n)$ est alors inférieure à $2 \cdot f(n)$, et on a bien $f \geq r$, $f \geq cr$, et $\forall n \in \mathbb{N}, f(n) \geq n^2$. $\square$

Lemme : Pour la mesure de compléxité $((\varphi_i)_{i\in\mathbb{N}}, (l_i)_{i\in\mathbb{N}})$, et pour toute fonction récursive totale $r : \mathbb{N} \to \mathbb{N}$, il existe une fonction récursive totale $\psi : \mathbb{N} \to \mathbb{N}$ telle que pour tout indice i pour ψ, il existe un indice j pour ψ tel que $r(c_j(n)) < c_i(n)$ pour presque tout n.

Preuve On peut supposer que la fonction r vérifie les hypothèses du précédent lemme, quitte à la remplacer par celle qu'il donne.
Soit $f : \mathbb{N} \to \mathbb{N}$ la fonction définie par : f(0) = 3, f(n+1) = r(f(n)) + 1.
On construit ensuite la fonction ψ, afin qu'elle vérifie les propriétés :

1. $\forall i \in \mathbb{N}$, si i est un indice pour ψ, alors $c_i(n) > f(n-i)$ pour presque tout n.
2. $\forall k \in \mathbb{N}, \exists j$ indice pour ψ, tel que $c_j(n) \leq f(n-k)$ pour presque tout n.

On déduit facilement de ces propriétés que ψ convient.
On construit une telle fonction ψ par récurrence sur n :

$$\psi(0) = \begin{cases} u(\varphi_0(0)) & \text{si } \; c_0(0) \leq f(0) \rightarrow \quad \text{marquer } 0 \\ 0 & \text{sinon} \end{cases}$$

$$\psi(n+1) = \begin{cases} u(\varphi_i(n)) & \text{où } i \text{ est le plus petit indice} \leq n \text{ non déja marqué} \\ & \quad \text{tel que } c_i(n) \leq f(n-i) \rightarrow \text{ marquer } i \\ 0 & \text{sinon} \end{cases}$$

La première condition est vérifiée. (de la même façon que dans la preuve du théorème 1).
Soit $k \in \mathbb{N}$. On souhaite construire une machine de Turing qui calcule ψ en utilisant un espace au plus $f(n-k)$. Soit s l'indice à partir duquel tous les indices $i \leq k$ qui seront marqués le sont. On peut supposer que l'on a précalculé les valeurs $\psi(n)$ pour $n \leq s$, ainsi que les indices marqués jusqu'à k (compris). La machine calcule ainsi $\psi(n)$ sans utiliser d'espace autre que celui utilisé pour le nombre n, pour $n \leq s$.
Pour $n > s$, le calcul de $\psi(n)$ s'éffectue de la façon suivante : pour n' allant de $s+1$ à n, pour i allant de $k+1$ à n', on éffectue :

1. on regarde si i est un indice déja marqué : si c'est le cas, on regarde l'indice i suivant et on recommence cette étape 1, sinon, on continue (on passe à l'étape 2)

2. on calcule $f(n'-i)$ (ce qui se fait en espace au plus $2 \cdot f(n'-i)$)
3. on calcule $c_i(n') \leq f(n'-i)$ (ce qui se fait en espace au plus $f(n'-i)$)
4. si on a $c_i(n') \leq f(n'-i)$, alors on calcule et on rend $\varphi_i(n')$ (ce qui n'utilise pas de place supplémentaire)
5. sinon, on continue le calcul pour le i suivant si $i < n'$, et si $i = n'$, on rend 0.

On recommence jusqu'à ce que n'=n, et on rend alors le résultat.
Le calcul de $\psi(n)$ se fait donc en un espace au plus

$$\underbrace{3 \cdot f(n-k-1)}_{\text{Calculs des } c_i(n') \leq f(n'-i)} + \underbrace{\log(n) \cdot (n-k)}_{\text{Stoquage des indices marqués}} \leq f(n-k) \text{ pour } n \text{ assez}$$

grand (car $\forall n \in \mathbb{N}, f(n) \geq n^2$ et $(f(n-1))^2 < f(n)$). $\square$

Théorème d'accélération de Blum : Pour toute mesure de complexité $((\varphi_i)_{i\in\mathbb{N}}, (c_i)_{i\in\mathbb{N}})$,et pour toute fonction récursive totale $r : \mathbb{N} \to \mathbb{N}$, il existe une fonction récursive totale $\psi : \mathbb{N} \to \mathbb{N}$ telle que pour tout indice i pour ψ, il existe un indice j pour ψ tel que $r(c_j(n)) < c_i(n)$ pour presque tout n.

Preuve On peut supposer que r est strictement croissante (quitte à prendre $r'(n) = n + \max_{i\leq n} r(i)$). Soit $f : \mathbb{N}^2 \to \mathbb{N}$ la fonction récursive donnée par le théorème 3. Soit $h(m) = m + \max_{n,m'\leq m} f(n, m')$. Soit ψ la fonction donnée par le lemme précédent, pour la fonction $horoh$. Soit i un indice pour ψ. D'après le lemme précédent, il existe un indice j pour ψ tel que :
$l_i(n) > h(r(h(l_j(n))))$ pour presque tout n. Les compléxités $l_i(n)$ sont supérieures à n, et on a $h(c_i(n)) \geq l_i(n) > h(r(h(l_j(n)))) \geq h(r(c_j(n)))$ par croissance de hor, pour presque tout n. Comme h est strictement croissante, on a finalement $c_i(n) \geq r(c_j(n))$, pour presque tout n. $\square$