Complexité et Récursion

Les fonctions récursives primitives présentées en PC 1 permettent de représenter des fonctions de $\mathbb{N}^n$ dans $\mathbb{N}$ qui terminent toujours, avec un langage dont l'expressivité est proche de celle des langages de programmation courants. La limitation principale permettant la terminaison porte sur la récursion. L'objectif de cette PC est de montrer qu'une restriction supplémentaire sur la récursion permet de caractériser exactement, et pourtant sans référence à un modèle de calcul particulier, la classe FP des fonctions qui peuvent être calculées par une machine de Turing déterministe en temps polynomial. (C'est cette même classe qui est utilisée pour la réduction $\leq$ entre problèmes).

L'ensemble des fonctions étudiées est défini par une induction très similaire à celles des fonctions récursives primitives, mais avec la restriction suivante : après un appel récursif, on n'a pas le droit de faire une récursion sur le résultat de l'appel. Pour imposer cela, les fonctions ont deux types d'arguments, dits *normaux* et *écartés*, les normaux apparaissant en premier et étant séparés des écartés par un point-virgule. Cette restriction impose que les fonctions ne peuvent grossir qu'avec modération.

Définition. Une fonction $f : \mathbb{N}^n \to \mathbb{N}$ appartient à la classe $\mathcal{P}$ si elle est : soit les constantes 0 ou 1 (alors $n = 0$), soit l'une des fonctions [1] :

- $\mathsf{BinSucc}_\ell$ les fonctions successeurs binaires pour $\ell \in \{0, 1\}$ qui calculent $x \mapsto 2x + \ell$: $\mathsf{BinSucc}_\ell(;0) = \ell$ et $\mathsf{BinSucc}_\ell(;a) = a\ell$ sinon ;
- $\mathsf{BinPred}$ la fonction prédecesseur binaire : $\mathsf{BinPred}(;0) = 0$ et $\mathsf{BinPred}(;a\ell) = a$ sinon ;
- $\mathsf{Proj}^i_{m,n}$: $(x_1, \ldots, x_m; x_{m+1}, \ldots, x_{m+n}) \mapsto x_i$ les fonctions de projection, pour $1 \leq i \leq m + n$;
- Cond, le branchement conditionnel défini sur $\mathbb{N}^3$:

$$\mathsf{Cond}(; a, b, c) = \begin{cases} b & \text{si } a \bmod 2 = 0, \\ c & \text{sinon} ; \end{cases}$$

- $\mathsf{Comp}_{m,p}(g, r_1, \ldots, r_m, s_1, \ldots, s_p)$ la composition

$$f(x_1, \ldots, x_n; a_1, \ldots, a_k) = g(r_1(x_1, \ldots, x_n;), \ldots, r_m(x_1, \ldots, x_n;); \\ s_1(x_1, \ldots, x_n; a_1, \ldots, a_k), \ldots, s_p(x_1, \ldots, x_n; a_1, \ldots, a_k))$$

  où les fonctions $g, r_1, \ldots, r_m, s_1, \ldots, s_p$ sont dans $\mathcal{P}$;
- $\mathsf{Rec}(g, h_0, h_1)$ la fonction définie par récursion comme

$$\begin{cases} f(0, x_2, \ldots, x_n; a_1, \ldots, a_k) = g(x_2, \ldots, x_n; a_1, \ldots, a_k), \\ f(x_1\ell, x_2, \ldots, x_n; a_1, \ldots, a_k) = h_\ell(x_1, \ldots, x_n; a_1, \ldots, a_k, f(x_1, \ldots, x_n; a_1, \ldots, a_k)), \quad \ell \in \{0, 1\}, \end{cases}$$

  où g, h_0, h_1 sont dans $\mathcal{P}$. Le cas où le premier argument vaut 1 est traité par le second choix, avec $x_1 = 0$ et $\ell = 1$.

1 Quelques fonctions simples avec des entiers unaires

Ces premières questions ont pour but de faire manipuler les règles de construction des fonctions de $\mathcal{P}$ et mieux percevoir la différence entre argument normal et argument écarté. Dans cette section, les arguments sont tous de la forme 1^n (le chiffre 1 répété n fois), avec $n \in \mathbb{N}$.

Question 1.1. *Montrer que la fonction* $\mathsf{UnaryAdd}(1^n; 1^m) = 1^{n+m}$ *est dans* $\mathcal{P}$.

1. Les entiers sont représentés par des mots binaires.

Question 1.2. *Montrer que si la fonction $f(x;y)$ est dans $\mathcal{P}$, alors la fonction $f' : (x,y;) \mapsto f(x;y)$ aussi.*

Question 1.3. *Montrer que la fonction* $\mathsf{UnaryMult}(1^n, 1^m;) = 1^{n\times m}$ *est dans* $\mathcal{P}$.

Question 1.4. *Si* M_1 *désigne la fonction* UnaryMult *et* $\mathsf{M}_{k+1} = \mathsf{Comp}_{2,0}(\mathsf{M}_k, \mathsf{M}_k, \mathsf{M}_k)$ *pour* $k > 1$, *montrer que la longueur de* $\mathsf{M}_k(1^n, 1^n;)$ *est polynomiale en* n, *pour tout* k.

Cette question montre que même en essayant très fort d'écrire des fonctions rapides, on a du mal à dépasser les polynômes. Il s'agit d'une limitation de $\mathcal{P}$, prouvée dans la section suivante.

2 Comportement polynomial

Pour mesurer la complexité, il sera commode d'utiliser la notation

$$|x| = \lceil \log_2(x+1) \rceil$$

pour représenter la taille binaire de x.

Question 2.1. *Montrer que pour tout $f \in \mathcal{P}$, il existe un polynôme q_f à coefficients positifs tel que $|f(x_1,\ldots,x_n;a_1,\ldots,a_k)| \leq q_f(|x_1|,\ldots,|x_n|) + \max_i |a_i|$ pour tout $(x_1,\ldots,x_n,a_1,\ldots,a_k)$.*

Question 2.2. *Montrer que les fonctions de $\mathcal{P}$ sont toutes calculables en temps polynomial.*

3 $\mathcal{P}$ et l'ensemble des fonctions calculables en temps polynomial

D'après la section précédente, les fonctions de $\mathcal{P}$ sont toutes calculables en temps polynomial. Nous allons maintenant prouver que réciproquement, toute fonction de $\{0,1\}^*$ dans $\{0,1\}^*$ se calculant en temps polynomial peut s'écrire comme une fonction de $\mathcal{P}$ n'ayant que des arguments normaux.

Par définition, une fonction f calculable en temps polynomial est une fonction pour laquelle il existe une machine de Turing T_f qui, prenant x en entrée, calcule $f(x)$ en un nombre d'étapes polynomial en $|x|$. Nous allons donc commencer par nous doter de quelques outils supplémentaires pour manipuler ce calcul. Une configuration de la machine de Turing sera donnée par un triplet :

— un entier qui marque le numéro de l'état ;
— un entier qui code la partie du ruban à gauche de la tête de lecture jusqu'au premier B ;
— un entier qui code la partie du ruban de la tête de lecture jusqu'au premier B à sa droite.

Pour les deux derniers entiers, les bits de poids faible sont les plus proches de la tête de lecture, et on considère l'entier obtenu en les préfixant d'un 1 pour éviter les ambiguïtés. Par exemple, la configuration où la machine est dans l'état numéro 4, et son ruban est

$$(B,1,0,\underline{0},1,0,B)$$

avec la tête de lecture au niveau du caractère souligné sera codé par les entiers $(4,6,10)$.

Une première étape consiste à exprimer une configuration comme un entier.

Question 3.1. *Exprimer par une fonction de $\mathcal{P}$ le calcul de la configuration initiale pour une fonction à un argument x, vu comme en entier avec la convention ci-dessus. (On pourra utiliser les fonctions τ_k définies dans la Section 4.2).*

Question 3.2. *Exprimer par une fonction de $\mathcal{P}$ le déplacement de la tête de lecture vers la droite, sans changer le caractère lu. (On pourra utiliser les fonctions de la Section 4.2).*

(Le déplacement vers la gauche s'effectue de la même façon et n'est pas demandé.)

À ce stade on admet qu'on peut écrire une fonction $\mathsf{Next}_M(x;)$ de $\mathcal{P}$ qui code la fonction de transition d'une machine de Turing M donnée. On souhaite ensuite écrire une fonction $\mathsf{Run}_M(y,x)$ qui calcule la configuration de la machine après $|y|$ étapes sur l'entrée x.

Question 3.3. *Pourquoi l'écriture*

$$\mathsf{Run}_M(0, x;) = \mathsf{Init}(x;), \quad \mathsf{Run}_M(y\ell, x;) = \mathsf{Next}(\mathsf{Run}_M(y, x;);)$$

ne convient-elle pas ?

Cette difficulté est contournée grâce au lemme suivant qui permet la composition avec certaines fonctions récursives primitives à croissance modérée.

Lemme 1. *Soient g et h dans $\mathcal{P}$ et soit f la fonction récursive primitive définie par*

$$f(0, y) = g(y;), \qquad f(x\ell, y) = h(f(x, y);).$$

S'il existe un polynôme P_f tel que $|f(x, y)| \leq P_f(|x|, |y|)$ pour tout (x, y), alors $(x, y;) \mapsto f(x, y)$ elle-même est dans $\mathcal{P}$.

Question 3.4. *Montrer à l'aide de ce lemme que si $f(x_1, \ldots, x_k)$ est une fonction calculable en temps polynomial, alors $f(x_1, \ldots, x_k;)$ est dans $\mathcal{P}$.*

Pour prouver le lemme, le principe est d'utiliser une récursion sur un entier qui sera ultérieurement borné par le nombre d'étapes utilisées par la fonction de $\mathcal{P}$ considérée. Dans un premier temps, on considère une fonction à un paramètre normal et une réciproque partielle à la question 1.2.

Question 3.5. *Montrer que pour toute fonction $f(x; a) \in \mathcal{P}$, il existe une fonction $\hat{f}(w; x, a)$ dans $\mathcal{P}$ et un polynôme p_f à coefficients positifs tels que $\hat{f}(w; x, a) = f(x; a)$ pour tout x et tout w avec $|w| \geq p_f(|x|, |a|)$.*

Question 3.6. *Prouver le lemme en construisant une fonction $\hat{f}$ de $\mathcal{P}$ telle que $\hat{f}(w, w; x, y) = f(x, y)$ pour tout (x, y) et tout w avec $|w| \geq p_f(|x|, |y|)$ pour un p_f à déterminer. (On pourra utiliser les fonctions de la section suivante.)*

La section qui suit définit dans des exercices un certain nombre de fonctions qui pourront être utilisées dans les sections précédentes (lorsque c'est explicitement indiqué !). Il est possible de résoudre ces exercices s'il reste du temps à la fin.

---

4 Fonctions utiles dans $\mathcal{P}$

Par rapport aux calculs que nous avions effectués avec des fonctions récursives primitives, il faut faire attention qu'ici les entiers sont écrits en binaire. Cette restriction est cruciale pour la complexité, mais complique un peu les preuves. Par exemple, montrer que l'addition est dans $\mathcal{P}$ sans utiliser le résultat de la section précédente est loin d'être immédiat. Bien entendu, il faut aussi faire attention aux arguments normaux et écartés, et s'assurer que les compositions sont toujours possibles et que les récursions n'utilisent leurs résultats intermédiaires que comme arguments écartés.

4.1 Décalage et test

Question 4.1. *Montrer que la fonction $\mathsf{Length}(x;) = 2^{|x|}$, ainsi que la fonction $\mathsf{Diff}(x, y;)$ qui vaut $2^{|y|-|x|}$ si $|y| \geq |x|$ et 0 sinon sont dans $\mathcal{P}$.*

Question 4.2. *Montrer que la fonction $\mathsf{RightShift} : (x; y) \mapsto y \operatorname{div} 2^{|x|}$ est dans $\mathcal{P}$.*

Question 4.3. *Montrer que la fonction $\mathsf{Test0}(x;)$ qui vaut 1 si $x = 0$ et 0 sinon, ainsi que la fonction de comparaison $\geq (x, y)$ qui vaut 1 si $|x| \geq |y|$ et 0 sinon sont dans $\mathcal{P}$.*

Question 4.4. *Montrer que la fonction $\mathsf{Bit}(x; y)$ qui calcule le $|x| + 1^e$ bit de y ($|x| \leq |y|$), c'est-à-dire $(y \div 2^{|x|}) \bmod 2$, est dans $\mathcal{P}$.*

4.2 Manipulations de chaînes

Question 4.5. *Montrer que la fonction* Reverse *qui prend un entier $x = (\ell_1, \ldots, \ell_m)_2$ en binaire et renvoie l'entier $(\ell_m, \ldots, \ell_1)_2$ est dans $\mathcal{P}$.*

Question 4.6. *Montrer que la fonction* $\mathsf{Concat}(x, y;) \mapsto x2^{|y|} + y$ *qui concatène les écritures binaires de x et y est dans $\mathcal{P}$.*

Question 4.7. *Montrer que la fonction $\tau : (x, y;) \mapsto (2^M + x) * (2^M + y)$, où $M = \max(|x|, |y|)$ et '$*$' représente la concaténation, est dans $\mathcal{P}$.*

Question 4.8. *À l'inverse, écrire deux fonctions π_1, π_2 de $\mathcal{P}$ pour extraire les entiers x, y de cette représentation.*

La fonction τ s'étend aux k-uplets en définissant $\tau_2(x, y;) = \tau(x, y;)$ puis $\tau_k(x_1, \ldots, x_k) = \tau(x_1, \tau_{k-1}(x_2, \ldots, x_k))$ pour $k > 2$. Les fonctions π_k d'extraction correspondantes sont clairement dans $\mathcal{P}$ et on ne demande pas de les écrire.

Références

[1] Stephen Bellantoni and Stephen Cook. A new recursion-theoretic characterization of the polytime functions. *Computational Complexity*, 2(2) :97–110, 1992. URL `http://dx.doi.org/10.1007/BF01201998`.