Master informatique, semestre 1
Modèles de calcul, devoir 2004

Ce devoir doit être rédigé *individuellement*, et remis à l'enseignant de votre groupe de TD, au plus tard le :

vendredi 17 décembre 2004, à 12h.

Il n'est pas nécessaire de tout traiter : ce travail ne constitue un entraînement utile que si vous *comprenez* tout ce que vous écrivez ! Un devoir partiel rédigé *avec soin* sera mieux noté qu'un devoir apparemment plus complet, mais brouillon.

Exercice 1

Un corrigé est disponible ci-dessous.

On sait (théorème de Rice) que l'ensemble F des procédures p qui calculent la fonction exponentielle de base 2 (autrement dit telles que, pour tout x : $p(x) = 2^x$), est indécidable ; cet exercice montre qu'il n'est pas récursivement énumérable, et son complémentaire non plus.

On suppose par l'absurde que F est récursivement énumérable, donc que sa fonction *semi-caractéristique* χ'_F est calculable. On appelle `scF` une procédure qui calcule χ'_F , et on considère la procédure `gamma` suivante (et une procédure auxiliaire pour calculer 2^x) :

```
int exp2 (int n) {
  int i, e = 1;
  for (i = 0; i < n; i++)
    e = 2 * e;
  return e;
}

int gamma (int x) {
  if (h (scF, gamma, x)) return 0;
  return exp2 (x);
}
```

On rappelle que h (p, x) désigne le prédicat (non calculable) qui exprime que le calcul de $p(x)$ termine, et que h (p, x, t) désigne le prédicat (calculable) qui exprime que le calcul de $p(x)$ est terminé au temps t.

1. Le prédicat h (`scF` , p) exprime une propriété de la procédure p : laquelle ? *Note* : on commencera par rappeler la définition de la fonction semi-caractéristique d'un ensemble.
2. Analyser soigneusement la fonction γ calculée par la procédure `gamma`. En déduire une contradiction, qui prouve que F n'est pas récursivement énumérable. *Note* : la réponse doit être argumentée avec précision et concision.
3. Montrer, le plus simplement possible, que l'ensemble complémentaire de F n'est pas récursivement énumérable.

Exercice 2 : machines de Turing

Dans cet exercice, l'alphabet ne comporte que deux caractères, le caractère blanc (noté b), et le caractère x ; la bande est infinie à gauche comme à droite, et elle est blanche lorsque la machine de Turing démarre (autrement dit sa valeur initiale est …*bbbbbbb*…). Les machines sont déterministes, et les règles sont notées comme dans le cours : par exemple la règle (q, b, q', x, D) indique que si, dans l'état q, la machine lit un blanc, elle passe dans l'état q', écrit un x, et la tête de lecture se déplace à droite (il y a toujours déplacement, à gauche ou à droite : il n'y a pas de troisième cas où la tête resterait immobile). Une machine s'arrête lorsqu'il n'y a pas de règle applicable.

Partie I.

Un corrigé est disponible ci-dessous.

Dans cette partie, les machines de Turing ne possèdent que deux états, notés 1 et 2 ; l'état 1 est l'état initial. On se propose d'énumérer explicitement *toutes* les machines de ce type qui *terminent*, et d'examiner leur comportement.

1. Les machines à examiner comportent au plus trois règles : expliquer pourquoi. Combien y a-t-il de machines comportant exactement trois règles ?

   Dans la suite, on ne s'intéresse qu'aux machines qui comportent exactement trois règles, et qui les utilisent toutes pendant leur fonctionnement (par exemple on ne s'intéresse pas à la machine qui s'arrête … immédiatement).

2. On souhaite énumérer seulement les machines essentiellement différentes ; en particulier, après avoir examiné une machine M, on ne s'intéresse pas à la machine symétrique M', obtenue en échangeant la gauche et la droite. Montrer que dans ces conditions, on peut convenir que la première règle est : (1, b, 2, x, D).

3. Compléter l'énumération demandée, de façon aussi rapide et intelligente que possible.

   Note : l'essentiel est de présenter *clairement* votre méthode d'énumération et d'analyse du comportement des machines ; si vous trouvez l'exercice trop long, vous pouvez interrompre l'énumération lorsque vous le jugez bon.

Partie II.

Un corrigé est disponible ci-dessous.

On analyse dans cette partie le comportement de la machine M, à cinq états, donnée par les neuf règles suivantes :

```
1 b 2 x G
2 b 3 x D
3 b 1 x G
4 b 1 x G
1 x 1 x G
2 x 2 x D
3 x 4 x D
4 x 5 x D
5 x 3 b D
```

Noter qu'il n'y a pas de règle pour l'état 5, lorsque le caractère lu est blanc : dans ce cas M s'arrête. Dans la suite, une configuration notée (u, q, v) indique que la machine est dans l'état q, que le mot uv est écrit sur la bande, et que la tête de lecture est positionnée sur le premier caractère de v.

1. Montrer qu'après 8 transitions, la machine est dans la configuration (x, 3, $xxxx$), notée en abrégé (x, 3, x^4).

2. On appellera "phase 1" la portion de calcul qui part de la configuration (x, 3, $x^{p(n)}$) ; M se déplace alors vers la droite en effaçant un caractère x sur trois ; la phase 1 se termine lorsque M passe dans l'état 1. Calculer alors, en fonction de $p(n)$, le nombre $c(n)$ de cellules comprises entre les caractères x extrêmes sur la bande, et le nombre $q(n)$ de "trous" (caractères blancs).

3. On appellera "phase 2" la portion de calcul qui suit une phase 1 : M se déplace vers la gauche pour "boucher les trous" (autrement dit remplacer un blanc isolé par un x), et ajoute deux x en fin de bande (à droite) pour chaque trou ainsi comblé ; la phase 2 se termine lorsqu'on atteint la configuration de départ d'une phase 1, et le compteur n est incrémenté. Calculer $p(n + 1)$ en fonction de $c(n)$ et de $q(n)$, puis directement en fonction de $p(n)$.

4. Calculer les valeurs successives de la suite $p(n)$, jusqu'à l'arrêt de la machine M (il est conseillé d'écrire un petit programme pour répondre à cette question). Quel est le test d'arrêt, exprimé sur $p(n)$? Combien de caractères x sont-ils écrits sur la bande en fin de calcul ?

Exercice 3 : fonction d'Ackermann

Un corrigé est disponible ci-dessous.

La fonction d'Ackermann est une fonction de deux variables entières $A\ (\ i, j\ )$; on peut considérer l'ensemble des valeurs de cette fonction comme un tableau infini (à deux dimensions), et on parlera ainsi de la *ligne numéro i* de A ; cette "ligne" est donc une fonction à un seul argument, qu'on notera f_i . Avec ces conventions, A est définie comme suit :

- f_0 est la fonction successeur ;
- la suite des valeurs de la ligne numéro $i + 1$ vérifie la récurrence : $u_j = f_i\ (\ u_{j-1}\ )$, avec par convention $u_{-1} = 1$.

Cette construction se traduit par les formules suivantes :

- pour tout j, $A\ (\ 0, j\ ) = j + 1$;
- pour tout i, $A\ (\ i + 1, 0\ ) = A\ (\ i, 1\ )$;
- pour tout i et pour tout j, $A\ (\ i + 1, j + 1\ ) = A\ (\ i, A\ (\ i + 1, j\ )\ )$.

1. Donner des formules explicites (c'est-à-dire sans récurrence) pour les fonctions f_1 , f_2 et f_3 .
2. Que vaut $A\ (\ 4, j\ )$ pour $j = 0, 1, 2$? Donner une formule "explicite" pour f_4 .
3. Que pensez-vous de $A\ (\ 5, 1\ )$?
4. On définit le λ-terme suivant :

   $$\mathbf{ack} = \lambda m\ .\ m\ (\ \lambda f\, n\ .\ n\, f\ (\ f\ \mathbf{1}\ )\ )\ \mathbf{succ}$$

   où **1** et **succ** sont les termes définis en cours pour représenter l'entier 1 et la fonction successeur. Montrer, en réduisant **ack 0**, puis **ack 1**, **ack 2** et **ack 3**, que ce λ-terme exprime bien la définition de la fonction d'Ackermann ; admirez la concision du λ-calcul !

Corrigé

Exercice 1

Voir l'énoncé ci-dessus.

1. Par définition d'une fonction semi-caractéristique, $\chi'_F\ (p)$ vaut 1 si la procédure p appartient à F, et n'est pas définie sinon. Donc le prédicat h (`scF` , p) exprime que la procédure p appartient à F, autrement dit calcule la fonction exponentielle (de base 2).

2. Supposons que la procédure `gamma` appartient à F : alors le calcul de `scF(gamma)` termine ; appelons τ la durée de ce calcul. On a donc :

   - $\gamma\ (x) = 2^x$ pour $x < \tau$
   - $\gamma\ (x) = 0$ pour $x \geqslant \tau$

   Ce qui contredit l'hypothèse `gamma` ∈ F.

   Supposons au contraire que la procédure `gamma` n'appartient pas à F : alors le calcul de `scF(gamma)` ne termine pas, et donc :

- $\gamma(x) = 2^x$ pour tout x

Ce qui contredit l'hypothèse `gamma` $\notin F$.

Dans les deux cas on a une contradiction, ce qui prouve que la procédure `scF` n'existe pas, autrement dit que la fonction χ'_F n'est pas calculable. Donc l'ensemble F n'est pas semi-décidable, et l'on sait qu'un ensemble est récursivement énumérable si et seulement s'il est semi-décidable ; donc F n'est pas récursivement énumérable.

3. Démontrer que le complémentaire de F, qu'on notera G, n'est pas récursivement énumérable, est plus simple ; il suffit de considérer la procédure suivante :

```
int gamma (int x) {
  scG (gamma);
  return exp2 (x);
}
```

Si `gamma` appartient à G, alors le calcul de `scG(gamma)` termine, donc $\gamma(x) = 2^x$ pour tout x, autrement dit `gamma` $\in F$.

Si au contraire `gamma` appartient à F, alors le calcul de `scG(gamma)` ne termine pas, donc pour tout x, $\gamma(x)$ est indéfini, et `gamma` $\in G$.

Dans les deux cas on a une contradiction, ce qui prouve que la procédure `scG` n'existe pas, et que G n'est pas récursivement énumérable.

Note : échanger `scF` et `scG`, ainsi que les `return` correspondants :

```
int gamma (int x) {
  if (h (scG, gamma, x)) return exp2 (x);
  return 0;
}
```

est une réponse *incorrecte*, car on n'aboutit à aucune contradiction en supposant que `gamma` appartient à G ; en effet, dans ce cas :

- $\gamma(x) = 0$ pour $x < \tau$
- $\gamma(x) = 2^x$ pour $x \geqslant \tau$

et `gamma` ne calcule pas la fonction exponentielle, conformément à la définition de G !

Exercice 2, partie I

Voir l'énoncé ci-dessus.

1. Comme les machines sont déterministes, elles comportent au plus quatre règles (deux choix pour l'état, deux choix pour le caractère lu). Mais une machine avec quatre règles ne peut pas terminer, avec la convention de terminaison employée ici (une machine s'arrête lorsqu'il n'y a pas de règle applicable).
   Note : on pourrait employer une autre convention de terminaison, en distinguant un état terminal, noté par exemple STOP, et distinct des états 1 et 2. Ces conventions sont équivalentes, sauf que la seconde permet à la machine d'effectuer une transition de plus avant de s'arrêter.

   Il faut choisir la règle absente (quatre choix), puis pour chacune des trois règles, il faut choisir le nouvel état (deux choix), le caractère écrit (deux choix), et le déplacement (deux choix) ; soit

$2^3 = 8$ choix pour chaque règle, et

$$4 * 8^3 = 2^{11} = 2048$$

machines à trois règles.

2. Initialement, la machine est dans l'état 1, et lit un blanc ; si la règle correspondante ne la fait pas changer d'état, la machine boucle, car après la première transition elle est toujours dans l'état 1 et lit à nouveau un blanc : elle part donc indéfiniment vers la droite (resp. vers la gauche) si la règle spécifie que la tête de lecture se déplace vers la droite (resp. vers la gauche).

   D'autre part, si la machine passe dans l'état 2 sans écrire un 'x' sur la bande, on se trouve dans la même situation que si l'état 2 était l'état initial. Enfin, par symétrie, on peut décider que la tête de lecture se déplace vers la droite, d'où un seul choix pour la première règle :

   $$(1, b, 2, x, \mathrm{D})$$

3. Après exécution de la première transition, la machine est dans l'état 2 et lit un blanc ; si la règle correspondante déplace à nouveau la tête de lecture à droite, la machine boucle (la tête de lecture se déplace indéfiniment vers à droite). Il reste donc quatre choix, qu'on examinera séparément.

   1. Règle (2, *b*, 1, *b*, G)

      Après deux transitions, la machine est dans l'état 1 et lit un *x*, qui est le seul caractère non blanc de la bande ; si la règle correspondante efface ce caractère, on se retrouve dans la configuration initiale (bande blanche, après un passage éventuel par l'état 2), et la machine boucle. Il reste donc seulement quatre règles à examiner pour la situation (1, *x*) (la machine est dans l'état 1 et lit un *x*) ; il n'y aura pas de règle pour la situation restante (2, *x*), qui servira à arrêter la machine.

      Les résultats sont résumés dans le tableau suivant ; la dernière colonne donne la suite des configurations, lorsque la machine termine, ou un cycle de configurations répété indéfiniment, lorsque la machine ne termine pas.

Règles	Terminaison	Configurations

`1, b, 2, x, D`
`2, b, 1, b, G`
`1, x, 1, x, D`
	Non	Ecrit indéfiniment *x* vers la droite :
		$(u, 1, b) \to (ux, 2, b) \to (u, 1, x) \to (ux, 1, b) \to$ etc.

`1, b, 2, x, D`
`2, b, 1, b, G`
`1, x, 1, x, G`
	Oui	$(b, 1, b) \to (x, 2, b) \to (b, 1, x) \to (b, 1, bx) \to (x, 2, x)$

`1, b, 2, x, D`
`2, b, 1, b, G`
`1, x, 2, x, D`
	Non	Oscille indéfiniment :
		$(x, 2, b) \to (b, 1, x) \to (x, 2, b) \to$ etc.

`1, b, 2, x, D`
`2, b, 1, b, G`
`1, x, 2, x, G`
	Non	Ecrit indéfiniment *bx* vers la gauche :
		$(x, 2, bu) \to (b, 1, xbu) \to (b, 2, bxbu) \to$
		$\to (b, 1, bbxbu) \to (x, 2, bxbu) \to$ etc.

   2. Règle (2, *b*, 1, *x*, G)

      Après deux transitions, la machine a écrit deux *x* consécutifs, est dans l'état 1 et lit le premier *x* ; il reste donc huit règles à examiner pour cette situation. Comme précédemment, la machine s'arrêtera si elle atteint la situation restante (2, *x*) (la machine est dans l'état 2 et lit un *x*).

Règles	Terminaison	Configurations
`1, b, 2, x, D`		Ecrit et efface indéfiniment *xx* en se déplaçant vers la droite :

2, b, 1, x, G 1, x, 1, b, D	Non	$(b, 1, b) \to (x, 2, b) \to (b, 1, xx) \to$ $\to (bb, 1, x) \to (bbb, 1, b) \to$ etc .
1, b, 2, x, D 2, b, 1, x, G 1, x, 1, b, G	Non	Ecrit indéfiniment x vers la gauche : $(b, 1, xx) \to (b, 1, bbx) \to (x, 2, bx) \to (b, 1, xxx) \to$ etc.
1, b, 2, x, D 2, b, 1, x, G 1, x, 1, x, D	Non	Ecrit indéfiniment xx vers la droite : $(b, 1, b) \to (x, 2, b) \to (b, 1, xx) \to$ $\to (x, 1, x) \to (xx, 1, b) \to$ etc.
1, b, 2, x, D 2, b, 1, x, G 1, x, 1, x, G	Oui	$(b, 1, b) \to (x, 2, b) \to (b, 1, xx) \to$ $\to (b, 1, bxx) \to (x, 2, xx)$
1, b, 2, x, D 2, b, 1, x, G 1, x, 2, b, D	Oui	$(b, 1, b) \to (x, 2, b) \to (b, 1, xx) \to (b, 2, x)$
1, b, 2, x, D 2, b, 1, x, G 1, x, 2, b, G	Oui	$(b, 1, b) \to (x, 2, b) \to (b, 1, xx) \to$ $\to (b, 2, bbx) \to (b, 1, bxbx) \to (x, 2, xbx)$
1, b, 2, x, D 2, b, 1, x, G 1, x, 2, x, D	Oui	$(b, 1, b) \to (x, 2, b) \to (b, 1, xx) \to (x, 2, x)$
1, b, 2, x, D 2, b, 1, x, G 1, x, 2, x, G	Oui	$(b, 1, b) \to (x, 2, b) \to (b, 1, xx) \to$ $\to (b, 2, bxx) \to (b, 1, bxxx) \to (x, 2, xxx)$

3. Règle (2, *b*, 2, *b*, G)

Après deux transitions, la machine est dans l'état 2 et lit un *x* isolé ; ce cas ressemble au cas 1, il faut examiner les règles pour la situation (2, *x*), et cette fois c'est la situation (1, *x*) qui servira à arrêter la machine. La symétrie n'est pas totale, car l'état 1 est initial, et nous avons fixé la règle de départ (1, *b*, 2, *x*, D) ; mais poursuivre l'énumération a peu d'intérêt, et il est raisonnable de l'arrêter là. Voici cependant les résultats.

Règles	Terminaison	Configurations
1, b, 2, x, D 2, b, 2, b, G 2, x, 1, x, D	Non	Ecrit indéfiniment x vers la droite : $(u, 1, b) \to (ux, 2, b) \to (u, 2, x) \to (ux, 1, b) \to$ etc.
1, b, 2, x, D 2, b, 2, b, G 2, x, 1, x, G	Oui	$(b, 1, b) \to (x, 2, b) \to (b, 2, x) \to$ $\to (b, 1, bx) \to (x, 2, x) \to (b, 1, xx)$
1, b, 2, x, D 2, b, 2, b, G 2, x, 2, x, D	Non	Oscille indéfiniment : $(b, 1, b) \to (x, 2, b) \to (b, 2, x) \to (x, 2, b) \to$ etc.
1, b, 2, x, D 2, b, 2, b, G 2, x, 2, x, G	Non	Part indéfiniment vers la gauche : $(b, 1, b) \to (x, 2, b) \to (b, 2, x) \to$ $\to (b, 2, bx) \to (b, 2, bbx) \to (b, 2, bbbx) \to$ etc.

4. Règle (2, *b*, 2, *x*, G)

Ce cas ressemble au cas 2, voir les commentaires ci-dessus (cas numéro 3).

Règles	Terminaison	Configurations
1, b, 2, x, D 2, b, 2, x, G	Oui	$(b, 1, b) \to (x, 2, b) \to (b, 2, xx) \to (b, 1, x)$

2, x, 1, b, D		
1, b, 2, x, D 2, b, 2, x, G 2, x, 1, b, G	Non	Ecrit indéfiniment x vers la gauche : $(b, 2, xx) \to (b, 1, bbx) \to (x, 2, bx) \to (b, 2, xxx) \to$ etc.
1, b, 2, x, D 2, b, 2, x, G 2, x, 1, x, D	Oui	$(b, 1, b) \to (x, 2, b) \to (b, 2, xx) \to (x, 1, x)$
1, b, 2, x, D 2, b, 2, x, G 2, x, 1, x, G	Oui	$(b, 1, b) \to (x, 2, b) \to (b, 2, xx) \to$ $\to (b, 1, bxx) \to (x, 2, xx) \to (b, 1, xxx)$
1, b, 2, x, D 2, b, 2, x, G 2, x, 2, b, D	Non	Ecrit indéfiniment x vers la gauche : $(b, 2, xx) \to (b, 2, x) \to (b, 2, b) \to$ $\to (b, 2, bx) \to (b, 2, bxx) \to (b, 2, bxxx) \to$ etc.
1, b, 2, x, D 2, b, 2, x, G 2, x, 2, b, G	Non	Ecrit indéfiniment x vers la gauche : $(b, 2, xx) \to (b, 2, bbx) \to (b, 2, bxbx) \to$ $\to (b, 2, bxxbx) \to (b, 2, bxxxbx) \to$ etc.
1, b, 2, x, D 2, b, 2, x, G 2, x, 2, x, D	Non	Ecrit indéfiniment xx vers la droite : $(b, 2, xx) \to (x, 2, x) \to (xx, 2, b) \to$ $\to (x, 2, xx) \to (xx, 2, x) \to (xxx, 2, b) \to$ etc.
1, b, 2, x, D 2, b, 2, x, G 2, x, 2, x, G	Non	Ecrit indéfiniment x vers la gauche : $(b, 2, xx) \to (b, 2, bxx) \to (b, 2, xxx) \to$ etc.

Exercice 2, partie II

Voir l'énoncé ci-dessus.

Dans une configuration (u, q, v), on note $u = b$ (resp. $v = b$) lorsque la bande est entièrement blanche à gauche de u (resp. à droite de v) ; on dira aussi, dans ce cas, que la tête de lecture est en "fin de bande".

1. Voici les premières configurations :
$$(b, 1, b) \to (b, 2, bx) \to (x, 3, x) \to (xx, 4, b) \to (x, 1, xx) \to$$
$$\to (b, 1, xxx) \to (b, 1, bxxx) \to (b, 2, bxxxx) \to (x, 3, xxxx).$$
2. On a :
$$c(n) = p(n) + 2$$
car en début de phase 1, la bande comporte $p(n) + 1$ caractères x consécutifs, et la dernière transition de la phase 1 ajoute un x en fin de bande.
$$q(n) = p(n) / 3 \text{ (quotient entier).}$$
3. On a :
$$p(n + 1) = c(n) + 2\ q(n) + 1$$
car chaque fois que la machine M "bouche un trou" (c'est-à-dire remplace un blanc isolé par un x), elle retourne vers la droite, dans l'état 2, pour ajouter deux x en fin de bande :
$$(u, 2, b) \to (ux, 3, b) \to (u, 1, xx)$$
D'autre part les dernières configurations de la phase 2 sont :
$$(b, 1, bu) \to (b, 2, bxu) \to (x, 3, xu)$$
d'où le "+1" final dans le calcul de $p(n + 1)$. En remplaçant $c(n)$ et $q(n)$ par leurs valeurs, on obtient :
$$p(n + 1) = p(n) + 2\ (\ p(n) / 3\ ) + 3$$
Attention : en principe, il faut effectuer la division entière par 3 avant de doubler le résultat ;

par exemple 2 (5 / 3) = 2, alors que le quotient entier 10 / 3 vaut 3. En fait on verra ci-dessous que lorsque $p(n)$ est congru à 2 modulo 3 — seul cas délicat pour le calcul de 2 ($p(n)$ / 3) —, la machine s'arrête (après exécution d'une dernière phase 1).

4. La récurrence ci-dessus, avec $p(1) = 4$, permet de calculer la suite des $p(n)$. La machine M s'arrête en fin de phase 1, lorsque $p(n)$ est congru à 2 modulo 3 : dans ce cas la phase 1 se termine dans l'état 5, et la machine lit un blanc, ce qui provoque son arrêt. Un petit programme donne la suite des valeurs de $p(n)$, accompagnées de leur classe modulo 3 :

```
[4, 1],    [9, 0],    [18, 0],   [33, 0],  [58, 1],
[99, 0],   [168, 0],  [283, 1],  [474, 0], [793, 1],
[1324, 1], [2209, 1], [3684, 0], [6143, 2]
```

Par miracle, il faut attendre $p(14)$ avant que la machine s'arrête ! La bande comporte alors :

$$6144 - (6143 /3) = 6144 - 2047 = 4097$$

caractères x. *Note* : cette machine, qui détient le record du "castor affairé" (busy beaver) pour les machines à 5 états, n'a pas été découverte par un humain, mais par un programme sophistiqué d'énumération, voir http://www.drb.insel.de/~heiner/BB/ .

Note : on aurait pu partir de $p(0) = 1$, qui correspond à la configuration obtenue après deux transitions.

Exercice 3 : fonction d'Ackermann.

Voir l'énoncé ci-dessus.

1. La ligne 1 débute par 2, puis on applique itérativement la fonction successeur ; donc :

$$f_1(n) = n + 2$$

La ligne 2 débute par 3, puis on passe d'un terme au suivant en ajoutant 2 ; donc :

$$f_2(n) = 2n + 3$$

Les premiers termes de la ligne 3 sont :

$$5 , f_2(5) = 13 , f_2(13) = 29 , f_2(29) = 61 , \text{etc.}$$

On conjecture donc :

$$f_3(n) = 2^{n+3} - 3$$

Vérification :

$$f_3(0) = 2^3 - 3 = 5$$
$$f_2(2^{n+3} - 3) = 2 (2^{n+3} - 3) + 3 = 2^{n+4} - 3 .$$

2. Les premiers termes de la ligne 4 sont :

$$13 = 2^4 - 3, f_3(13) = 2^{16} - 3 , f_3(2^{16} - 3) = 2^{2^{16}} -3 , \text{etc.}$$

$f_4(n)$ est une "tour d'exponentielles", de hauteur $n + 3$ (tous les nombres de la tour sont égaux à 2), à laquelle on retranche 3 ; on vérifie facilement qu'une telle suite vérifie bien la récurrence $u_{n+1} = f_3(u_n)$.

3. $A(5, 0) = 2^{16} - 3 = 65533$, donc $A(5, 1)$ est une tour d'exponentielles de hauteur … 2^{16}. La fonction d'Ackermann illustre la possibilité de définir des nombres défiant toute imagination par un programme très court (il est très facile de transformer la définition de la fonction d'Ackermann en un programme de calcul), évidemment impossible à exécuter en pratique : on n'ose envisager ce que vaut $A(10, 10)$!

4. Notons $M = \lambda f n . n f (f \mathbf{1})$, de telle sorte que :

$$\mathbf{ack} = \lambda m . m M \mathbf{succ}$$

On a alors :

$$\mathbf{ack\ 0} \rightarrow \mathbf{0}\ M\ \mathbf{succ} \rightarrow \mathbf{succ}\ (\text{car } \mathbf{0} \text{ ignore son premier argument})$$

$$\mathbf{ack\ 1} \to \mathbf{1}\ M\ \mathbf{succ} \to M\ \mathbf{succ} \text{ (car } \mathbf{1} \text{ est l'identité)} \to$$
$$\to \lambda\, n\, .\, n\ \mathbf{succ}\ (\ \mathbf{succ}\ \mathbf{1}\ ) \to \lambda\, n\, .\, n\ \mathbf{succ}\ \mathbf{2}$$

et ceci est bien la fonction $f_1 : n \to n + 2 = \text{succ}^n(2)$. Par exemple :

$$\mathbf{ack\ 1\ 3} \to \mathbf{succ}\ (\mathbf{succ}\ (\mathbf{succ}\ \mathbf{2})) \to \mathbf{5}.$$

Attention : ne pas réduire "$\lambda\, n\, .\, n\ \mathbf{succ}\ \mathbf{2}$" en "$\lambda\, n\, .\, n\ \mathbf{3}$", car :

$$\lambda\, n\, .\, n\ \mathbf{succ}\ \mathbf{2} = \lambda\, n\, .\, (\ n\ \mathbf{succ}\ )\ \mathbf{2} \neq \lambda\, n\, .\, n\ (\ \mathbf{succ}\ \mathbf{2}\ ).$$

Calcul de **ack 2** :

$$\mathbf{ack\ 2} \to \mathbf{2}\ M\ \mathbf{succ} \to M\ (\ M\ \mathbf{succ}\ ) \to \lambda\, n\, .\, n\ (\ M\ \mathbf{succ}\ )\ (\ M\ \mathbf{succ}\ \mathbf{1}\ )$$

Or "M **succ**" est la fonction **ack 1**, qu'on applique itérativement $n + 1$ fois à l'entier **1**, ce qui est exactement la définition de la "ligne 2" de A. De même :

$$\mathbf{ack\ 3} \to \mathbf{3}\ M\ \mathbf{succ} \to M\ (M\ (\ M\ \mathbf{succ}\ )\ ) = M\ (\ \mathbf{ack\ 2}\ ) \to \lambda\, n\, .\, n\ (\ \mathbf{ack\ 2}\ )\ (\ \mathbf{ack\ 2\ 1}\ )$$

C'est la fonction qui applique itérativement la fonction "**ack 2**" $n + 1$ fois à l'entier **1**. Et ainsi de suite…

Note. On comprend maintenant comment a été construit **ack** :

$$M = \lambda f\, n\, .\, n\, f\, (\, f\ \mathbf{1}\ )$$

est l'opérateur qui transforme f en u, avec (en notant u comme une suite) :

$$u_n = f^{n+1}(1)$$

et donc u vérifie la récurrence :

$$u_0 = f(1)\ ,\ u_{n+1} = f\,(\ u_n\ )$$

Ensuite

$$\mathbf{ack} = \lambda m\, .\, m\ M\ \mathbf{succ}$$

itère m fois la construction précédente (transformation de f en u), à partir de la fonction *successeur* : c'est exactement la définition de la fonction d'Ackermann, donnée en début d'énoncé. La concision et la puissance d'expression du λ-calcul viennent du principe qu'un entier m est vu comme l'opérateur "itérer m fois", et que cet opérateur peut être appliqué à n'importe quel autre opérateur (pas seulement aux fonctions $f : \mathbf{N} \to \mathbf{N}$) ; ici on itère m fois M, qui transforme une fonction en une autre fonction (notée ci-dessus comme une suite, mais suite et fonction $\mathbf{N} \to \mathbf{N}$ désignent le même objet, seule la notation change).