Gribouillis sur la calculabilité supérieure

David A. Madore

22 janvier 2010

CVS :
`$Id: calculabilite.tex,v 1.2 2010-01-22 20:49:01 david Exp $`

Je me suis intéressé ces derniers temps à un domaine qu'on pourrait appeler la « calculabilité supérieure » (en anglais, « higher recursion theory ») : en une ligne, il s'agit d'étudier des notions de calculabilité qui sont au-delà (voire bien au-delà) de la calculabilité au sens de Church-Turing, et éventuellement d'inventer des modèles de « machines » généralisées qui réalisent ces notions de calculabilité supérieure. Pour vérifier ma compréhension du domaine, je vais écrire des (« des » = au moins un, celui-ci :-) gribouillis pour former une introduction à quelques facettes de ce domaine.

Ce gribouillis-ci concerne les fonctions et ensembles hyperarithmétiques, et les « machines hyperarithmétiques », et essaie d'expliquer pourquoi c'est une classe très naturelle et élégante, peut-être même plus naturelle que les fonctions récursives (=calculables au sens de Church-Turing). J'en profite pour évoquer des sujets connexes au passage (machines de Turing avec oracles et avec hyperoracles).

J'ai essayé de garder des prérequis minimaux (même si je ne sais pas très exactement ce qu'ils sont), voire de faire de la « vulgarisation ». Je ne sais pas dans quelle mesure j'aurai réussi, mais au moins, comme ce gribouillis est très long, je vais essayer de faire en sorte qu'il soit facilement lisible en diagonale, et que les sous-parties qui le constituent soient intéressantes indépendamment les unes des autres (donc n'hésitez pas à sauter les bouts qui vous semblent ennuyeux !). Notamment, je donne plusieurs caractérisations des ensembles hyperarithmétiques, parce que si on ne trouve pas que « Δ^1_1 » est une jolie notion, on peut préférer les machines hyperarithmétiques. Plus précisément, le fil conducteur de ce gribouillis, c'est l'équivalence, pour un ensemble A d'entiers naturels entre les différentes notions suivantes d'« hyperarithmétique » que je vais définir ci-dessous :

- A est Δ^1_1 pour la hiérarchie analytique,
- A est calculable par une machine de Turing ayant un hyperoracle permettant de décider si une fonction $\mathbb{N} \to \mathbb{N}$ prend une valeur donnée,
- le degré de Turing de A est majoré par ω_1^{CK} itérations du saut de Turing,

– A est calculable par une machine ordinale qui peut effectuer des « grandes boucles » de taille ω_1^{CK},
– A appartient à $L_{\omega_1^{\mathrm{CK}}}$ où L_α désigne la α-ième étape de l'univers constructible de Gödel.

J'essaierai de faire en sorte que les éventuels griboullis suivants soient autant que possible indépendants de celui-ci.

1 Fonctions générales récursives et primitives récursives, problème de l'arrêt

Une fonction (générale) récursive (ou simplement « calculable »), c'est une fonction calculable par n'importe lequel des procédés mathématiques équivalents pour définir une fonction « mécaniquement » (ou : « effectivement ») calculable. Par exemple, les machines de Turing. On peut par exemple les définir comme la plus petite classe de fonctions partielles (prenant en entrée des tuples de taille fixe d'entiers naturels et renvoyant un entier naturel) contenant les constantes et les projections (identités), les opérations arithmétiques de base (disons) et le test (genre, une fonction de (x, y, u, v) qui renvoie u si $x = y$ et v sinon), et stable par composition, par récursion primitive (si $g(\ldots)$ est récursive et $h(\ldots)$ est récursive, alors $f(n, \ldots)$ définie par $f(0, \ldots) = g(\ldots)$ et $f(n+1, \ldots) = h(f(n, \ldots), \ldots)$ est encore récursive) et par schéma μ (si $f(n, \ldots)$ est récursive, alors $\mu f(\ldots)$ défini comme le plus petit n tel que $f(n, \ldots) = 0$ et que tous les $f(k, \ldots)$ pour $k \langle n$ soient définis, est récursive). Dans tout ça, il faut évidemment garder en tête que les fonctions peuvent ne pas être définies, correspondant au fait que le programme qui les calcule ne termine pas forcément.

Informellement, les fonctions récursives sont celles calculées par à peu près n'importe quel langage de programmation idéalisé pas trop con, par exemple le langage « FlooP » présenté dans *Gödel, Escher, Bach* ou essentiellement tout langage de programmation réel rendu idéal. (Note : dans ma définition ci-dessus, mes fonctions n'ont pas accès à une mémoire ; en fait, ça ne change rien de leur en donner un, parce qu'on peut toujours coder des suites finies d'entiers naturels, donc un état quelconque d'une mémoire, sous la forme d'un unique entier naturel, au moyen d'« opérations arithmétiques de base ». Si on imagine un langage de programmation, c'est sans doute mieux d'imaginer qu'il peut allouer des tableaux ou quelque chose comme ça.)

Un point très important avec les fonctions générales récursives, c'est qu'on peut les numéroter de façon standard par des entiers naturels de façon à pouvoir trouver une fonction « universelle » $u(n, k)$ telle que $u(n, \langle k_1, \ldots, k_r \rangle) = s$ ssi la n-ième fonction récursive appliquée aux arguments $k_1, \ldots, k_r$ (est définie et) vaut s. (Ici, $\langle \ldots \rangle$ désigne un codage standard des r-uples d'entiers naturels par des entiers naturels.) Au niveau langages de programmation, ça veut dire qu'on peut écrire un interpréteur : un truc qui

va prendre un programme n et des arguments k, et simuler l'exécution du programme sur ces arguments. Une autre chose qu'il est utile de noter, c'est qu'on peut toujours supposer qu'une fonction récursive a accès a son propre numéro comme argument (c'est-à-dire, au niveau langage de programmation, qu'un programme peut lire son propre code source) : c'est le principe des « quines » en programmation ; formellement, si h est une fonction récursive, il existe n tel que $u(n, \langle k_1, \ldots, k_r \rangle) = h(n, k_1, \ldots, k_r)$.

Parmi les fonctions générales récursives, il y en a un sous-ensemble assez important (et qui a été défini, historiquement, avant les fonctions générales récursives), ce sont les primitives récursives. En gros, ce sont celles calculées par un langage de programmation qui n'a pas le droit de boucler indéfiniment : on ne peut pas faire d'appel de fonction récursif (une fonction ne peut appeler qu'une fonction définie strictement avant), et chaque boucle doit être bornée à l'avance (quand je dis borné, je veux dire par une variable préalablement calculée : on ne demande pas que ce soit par une constante dans le programme). Un exemple détaillé de tel langage est le langage « BlooP » de *Gödel, Escher, Bach*. Sinon, mathématiquement, il suffit de retirer le schéma μ de ma définition précédente. Les fonctions sont alors forcément totales : une fonction primitive récursive est définie partout, c'est-à-dire que le programme correspondant termine toujours.

En gros, il faut considérer les fonctions primitives récursives comme une classe de complexité algorithmique : une fonction $n \mapsto f(n)$ est primitive récursive si et seulement si elle est calculable par une machine de Turing en moins de $t(n)$ étapes où t est... une fonction primitive récursive, justement. Un exemple de fonction qui n'est pas primitive récursive mais néanmoins générale récursive et totale (et clairement calculable mécaniquement), c'est la fonction universelle u des fonctions primitives récursives : elle ne peut pas être primitive récursive par un argument diagonal évident (#1) : si la fonction $(n, k) \mapsto u(n, k)$ était primitive récursive alors $(n, k) \mapsto u(n, k) + 1$ le serait aussi, elle aurait un numéro m, et alors $u(m, m) = u(m, m) + 1$, une contradiction. Un autre exemple, c'est une fonction à croissance trop rapide, comme la fonction d'Ackermann. Notons cependant (#2) que les fonctions primitives récursives peuvent faire tout ce que peuvent faire les machines de Turing si on « leur donne le temps » : le calcul des m premières étapes de l'exécution de la machine de Turing numéro n est primitif récursif, donc pour toute fonction générale récursive $f(n, \ldots)$ il existe une fonction primitive récursive $F(k, n, \ldots)$ telle que pour tout k, soit $F(k, n, \ldots) = \text{`failed'}$, soit $F(k, n, \ldots) = f(n, \ldots)$, ce dernier cas se produisant dès que $k \geq k_0$ avec k_0 dépendant de f et des arguments de celle-ci.

On pourrait construire des classes intermédiaires entre fonctions primitives récursives et générales récursives en admettant des conditions d'arrêt plus générales que « la boucle va faire au plus tant de tours ». Le bon cadre pour ça, ce serait les notations ordinales à la Kleene que je définirai plus bas (une fonction primitive récursive généralisée par une notation ordinale o, ce serait en gros une fonction qui a le droit de faire des boucles tant qu'elle décroît dans o) ; mais les fonctions p.r. ne sont pas trop le

sujet de ce post, donc je laisse ça de côté.

Une partie S de $\mathbb{N}$ est dite récursive lorsque sa fonction indicatrice est générale récursive, c'est-à-dire concrètement lorsqu'on peut écrire un programme qui prend un entier en entrée, termine toujours, et renvoie 0 ou 1 selon que le nombre appartient à S ou pas. On peut définir de même la notion de partie primitive récursive de $\mathbb{N}$ (par exemple, l'ensemble des n tels que la n-ième fonction primitive récursive vaille 0 en 0 est une partie récursive non primitive récursive de $\mathbb{N}$).

Une partie S de $\mathbb{N}$ est dite « récursivement énumérable » (abrégé en « r.é. »), ou « semi-récursive », lorsque c'est l'ensemble des points où une fonction générale récursive partielle est définie ou, de façon équivalente, l'image d'une fonction générale récursive partielle ou même totale (c'est la même chose, parce qu'on peut simuler l'exécution diagonale de la fonction f sur tous les entiers naturels, et renvoyer successivement les valeurs pour lesquelles f a terminé correctement) ; c'est aussi la même chose qu'une partie qui est l'image d'une fonction primitive récursive (à cause de l'argument (#2) ci-dessus).

Une partie S de $\mathbb{N}$ est récursive si et seulement si elle est récursivement énumérable et son complémentaire aussi (on s'en convainc facilement, parce que pour tester si un entier est dedans, on énumère S et son complémentaire jusqu'à décider la question). Dans le même genre, une fonction *partielle* est récursive ssi son graphe est une partie récursivement énumérable, mais une fonction *totale* est récursive ssi son graphe est une partie récursive, ou simplement récursivement énumérable.

L'argument diagonal (#1) ci-dessus (montrant qu'il n'existe pas de fonction primitive récursive universelle pour celles-ci) ne marche pas contre les fonctions générales récursives, parce que $u(m, m)$ est simplement non défini. Par contre, ce qu'il montre, c'est que la fonction (totale) suivante n'est pas récursive : $h(n, k) = 1$ si la n-ième fonction générale récursive est définie sur l'entrée k (si la n-ième machine de Turing s'arrête sur k, quoi), et 0 sinon. Cette fonction h s'appelle la fonction d'arrêt des machines de Turing, et le fait que h ne soit pas récursive s'appelle l'indécidabilité du problème de l'arrêt. Plus exactement, ceci fournit un exemple de partie de $\mathbb{N}$ (ou de $\mathbb{N}^2$, 'fin peu importe) qui n'est pas récursif : l'ensemble des (n, k) tels que la n-ième machine de Turing s'arrête sur l'entrée k ; elle est pourtant récursivement énumérable, parce qu'on peut toujours simuler en parallèle l'exécution de toutes les machines de Turing (énumérer les étapes diagonalement), et dire coucou quand il y en a une qui s'arrête.

Un des thèmes de ce qui suit, ça va être de se demander ce qui se passe si on ajoute magiquement à une machine de Turing le pouvoir de décider de l'arrêt des machines de Turing ; puis de décider l'arrêt de ces machines étendues ; puis de ces machines étendues de nouveau ; « et ainsi de suite ». Mais pour définir comment on peut « donner un pouvoir » à une machine de Turing, il faut d'abord que je parle d'oracles.

2 Machines de Turing avec oracles, relativisation

Une machine de Turing avec oracle, c'est une machine de Turing qui a le droit, à n'importe quel point de son exécution, de « consulter l'oracle » qu'on lui fournit. Plus formellement, si O (l'oracle) est une partie de $\mathbb{N}$ (vue comme sa fonction indicatrice) ou une fonction $\mathbb{N} \to \mathbb{N}$ (totale), on définit les fonctions générales récursives *relativement à* O (ou « calculables à partir de O », ou « Turing-réductibles à O ») de la même manière que les fonctions générales récursives sauf qu'en plus on impose autoritairement que O soit dans la classe. Donc concrètement, le programme peut interroger O (autant de fois qu'il veut). Évidemment, pour chaque O il y a une notion différente de machines avec oracle O ; il est cependant utile de remarquer qu'elle est uniforme (la numérotation des machines avec oracle ne dépend pas de quel est O : notamment, il existe un entier u tel que pour tout O la u-ième machine avec oracle O soit universelle pour les machines avec oracle O). On définit de façon évidente la notion de partie de $\mathbb{N}$ qui soit récursive relativement à O, ou récursivement énumérable relativement à O.

Il est pertinent de remarquer que, pour un calcul donné, une machine de Turing ne va jamais interroger qu'un nombre fini de valeurs de O. C'est quelque chose qu'il faut bien garder à l'esprit à chaque fois qu'il est question d'oracles.

Certains oracles n'apportent rien au niveau puissance de calcul : si O est une fonction constante, ou simplement récursive (dans l'absolu), alors une fonction est récursive relativement à O ssi elle est récursive tout court — c'est complètement évident. Autres remarques du même acabit : toute fonction est récursive relativement à elle-même ; et si une fonction H est récursive relativement à une fonction G elle-même récursive relativement à F, alors H est récursive relativement à F.

Ceci conduit à définir la notion de « degré de Turing » : le degré de Turing d'une fonction totale O, c'est l'ensemble de toutes les fonctions totales O' telles que O' soit récursive relativement à O et réciproquement. Autrement dit, on prend le préordre de réductibilité de Turing, et on considère la relation d'équivalence qu'il définit : c'est l'équivalence de Turing, dont les classes sont les degrés de Turing. Là j'ai défini ça pour les fonctions (totales) : pour les parties de $\mathbb{N}$, on les identifie à leur fonction indicatrice (donc ce qui importe est d'être récursif relativement à un oracle, pas récursivement énumérable relativement à lui) ; on vérifie facilemnt que tout degré de Turing contient une fonction indicatrice (par exemple parce que toute fonction peut être calculée à partir de l'indicatrice de son graphe...) donc on peut aussi bien considérer les degrés de Turing comme des classes de parties de $\mathbb{N}$.

Parfois il est préférable de définir les choses un peu différemment, et de considérer que l'oracle n'est pas fixé mais fait partie des entrées : on peut donc dire qu'une fonction F de $\mathbb{N}^{\mathbb{N}}$ vers $\mathbb{N}$ (par exemple) est calculable (=récursive) lorsqu'il y a une machine de Turing avec oracle non spécifié qui quand on l'exécute sur un oracle f renvoie $F(f)$. Bien sûr, on peut aisément admettre plusieurs entrées, ou définir la notion de fonction de $\mathbb{N}^{\mathbb{N}}$ vers $\mathbb{N}$ calculable relativement à un oracle (pour l'instant dans $\mathbb{N}^{\mathbb{N}}$, je parlerai plus

loin des « hyperoracles » qui sont dans $\mathbb{N}^{\mathbb{N}^{\mathbb{N}}}$).

3 Degrés de Turing, saut de Turing

Les degrés de Turing forment un ensemble partiellement ordonné pour la relation « être calculable à partir de ». Cet ensemble a un plus petit élément, noté 0 : c'est le degré formé de toutes les fonctions totales récursives. Deux degrés de Turing a et b ont toujours une borne supérieure $a \vee b$, qui peut s'expliciter comme ceci : on prend des fonctions A et B représentant a et b, et on définit $C(2n) = A(n)$ et $C(2n+1) = B(n)$; alors le degré c de C est la borne supérieure de A et B. (On pourrait croire que ceci se généralise à une suite de degrés, mais il n'en est rien. Je reviendrai là-dessus.)

Il n'y a pas de plus grand élément : il est évident pour des raisons de cardinalité qu'aucun oracle ne permet de calculer toutes les fonctions $\mathbb{N} \to \mathbb{N}$, mais en fait on peut dire mieux : pour chaque degré de Turing d, il existe un degré strictement supérieur à d, à savoir le degré noté d' ou $\mathrm{TJ}(d)$ du problème de l'arrêt relativement à d (le fait que le problème de l'arrêt relativement à un oracle D ait un degré strictement supérieur à celui de D est assez évident — ce qui est un peu moins évident, c'est que ce degré ne dépend pas du choix de D dans d).

Cette opération $\mathrm{TJ}\colon d \mapsto d'$ s'appelle le « saut de Turing » et définit une fonction croissante des degrés de Turing vers les degrés supérieurs ou égaux à $0'$ (qui est donc le degré du problème de l'arrêt absolu). Il se trouve (mais c'est loin d'être évident) que l'image du saut de Turing est l'ensemble des degrés supérieurs ou égaux à $0'$: cela résulte du théorème suivant : pour tout degré d supérieur ou égal à $0'$, il existe un degré a tel que $a' = d$ et $a \vee 0' = d$ (théorème d'inversion de Friedberg).

On a donc en particulier défini le degré $0'$ (degré du problème de l'arrêt absolu), le degré $0''$ (degré du problème de l'arrêt pour les machines qui disposent de l'oracle de l'arrêt absolu, i.e., problème de l'arrêt de deuxième niveau), le degré $0'''$, et pour tout entier naturel r le degré $0^{(r)}$. Je vais dire dans un instant la difficulté qui se pose ensuite. Pour l'instant je reviens un peu sur la structure des degrés de Turing.

Il faut bien se garder de croire que d' est le plus petit degré strictement supérieur à d : il existe toujours des degrés strictement intermédiaires entre d et d'. Il faut aussi se garder de croire que les degrés sont totalement ordonnés : il existe des degrés incomparables (en fait, il existe un degré incomparable à tout degré donné, sauf 0). Il faut aussi se garder de croire que le saut de Turing est strictement croissant (whatever that means) : on peut très bien avoir $a' = b'$ alors que $a < b$ (et même, pour tout a un tel b existe), et par ailleurs on peut avoir $a' = b'$ ou $a' < b'$ alors que a et b sont incomparables. Il faut se garder de croire qu'il existe toujours des degrés intermédiaires entre deux degrés donnés : il existe des degrés minimaux, c'est-à-dire tels que $m > 0$ mais qu'il n'y ait aucun degré strictement entre 0 et m (et même, des degrés minimaux au-dessus de n'importe quel degré fixé).

Un degré est dit « récursivement énumérable » lorsqu'il contient une partie récursivement énumérable de $\mathbb{N}$ (évidemment, à part 0, il contiendra toujours des ensembles non-r.é., par exemple le complémentaire d'un ensemble r.é. non récursif). Par exemple, le degré $0'$ est récursivement énumérable, et on se convainc facilement que tout degré récursivement énumérable est $\leq 0'$. La réciproque, cependant, n'est pas vraie : il existe des degrés $\leq 0'$ qui ne sont pas récursivement énumérables (Schoenfield). D'autre part, il existe des degrés récursivement énumérables strictement entre 0 et $0'$ (Friedberg-Muchnik : résolution du « problème de Post »). On peut bien sûr aussi définir un degré r.é. relativement à un degré d donné (et alors le degré est $\leq d'$).

Ce sont deux sujets assez différents, en fait : la théorie des degrés de Turing et la théorie des degrés de Turing récursivement énumérables. Elles n'ont pas vraiment la même saveur. Par exemple, l'ordre sur les degrés n'est pas dense (il y a des degrés minimaux), par contre l'ordre sur les degrés r.é. est dense. Mais dans chacun de ces sujets on a un nombre absolument impressionnant de résultats sur l'ordre sur les degrés, le saut de Turing, le plongement de structures ordonnées ou munies d'inf/sup dedans, l'existence d'automorphismes, etc. Personnellement je m'y perds complètement et j'ai l'impression d'être « lost in a maze of twisty little theorems, all alike ». Pour une illustration, voir l'article Wikipédia « Turing degree », le chapitre 13 (et notamment la liste d'énoncés page 294) du Rogers, *Theory of recursive Functions and Effective Computability* ; ou pour des choses vraiment détaillées, le livre de Lerman, *Degrees of Unsolvability* et l'article de Soare, *Recursively Enumerable Sets and Degrees*.

Mon but n'est *pas* de parler de ces résultats sur la structure d'ordre sur les degrés de Turing ou choses de ce genre. C'est plutôt de voir ce qui se passe si on essaie de « monter très haut » dans les degrés de Turing.

Je mentionne cependant un résultat fondamental qui aura éventuellement des conséquences plus tard si j'ai la patience d'aller jusque là. Si I est un idéal dénombrable des degrés de Turing, c'est-à-dire un ensemble dénombrable de degrés stable par réduction de Turing (si $a \leq b$ avec b dans I alors a est dans I), on peut se demander si I est un idéal principal (i.e., $\{a$ tels que $a \leq d\}$ pour un certain degré d) : ce n'est pas forcément le cas (ça c'est vraiment facile), mais par contre il existe toujours une « paire exacte » pour I, c'est-à-dire deux degrés d_1 et d_2 tels que I soit l'ensemble des degrés a vérifiant $a \leq d_1$ *et* $a \leq d_2$.

4 Le ω-saut de Turing

Imaginons qu'on ait une suite a_m de degrés représentés par des fonctions $F(m, \bullet)$. Il est assez évident que si on regarde la fonction F comme fonction de deux variables, son degré majore chacun des a_m. On pourrait croire que (comme on a défini la borne supérieure de deux degrés $a \vee b$) ceci fabrique la borne supérieure des a_m. Il n'en est rien, pour deux raisons fondamentales. La première, c'est que la fonction F peut avoir

un degré arbitrairement élevé pour une raison idiote : je peux toujours changer la valeur en 0 de chacune de mes fonctions $F(m, \bullet)$, ça va rester dans le même degré a_m, et du coup je peux réussir à cacher n'importe quelle fonction $F(\bullet, 0)$ dans le degré de F. La deuxième raison, c'est que si a_m est une suite strictement croissante, par exemple, il n'y a *jamais* de borne supérieure à l'ensemble des a_m : en effet, l'ensemble I des degrés qui sont inférieurs à l'un des a_m constitue ce que j'ai appelé ci-dessus un idéal dénombrable de degrés, donc il existe deux degrés d_1 et d_2 (une « paire exacte ») tels qu'un degré c soit inférieur à l'un des a_m ssi il est $\leq d_1$ et $\leq d_2$, et alors il est assez évident qu'une borne sup ne peut pas exister (car elle serait forcément $\leq d_1$ et $\leq d_2$).

Tout ceci est de mauvais augure si le but est de définir un degré $0^{(\omega)}$ qui vienne « naturellement » après la suite de degrés formés par les sauts finis 0, $0'$, $0''$, $0'''$, $0^{(4)}$, etc.

Bien que ça semble mal parti, on va quand même définir $0^{(\omega)}$ de la façon naïve : on prend le représentant de $0^{(r)}$ pour chaque r donné par le problème de l'arrêt relativement au précédent, c'est-à-dire que par récurrence sur r, H_r est l'ensemble des numéros des machines de Turing qui s'arrêtent quand on leur donne H_{r-1} comme oracle, et $0^{(\omega)}$ est le degré de Turing de l'ensemble des couples $\langle r, n\rangle$ tels que n soit dans H_r. Concrètement, $0^{(\omega)}$ est le degré qui permet de décider si une machine de Turing avec r niveaux d'oracles d'arrêt s'arrête, pour un r fixé. Un des thèmes qui va m'intéresser, c'est d'expliquer pourquoi ce degré est quand même naturel.

Je vais noter une caractérisation de $0^{(\omega)}$, mais qui n'est pas forcément super intuitive : c'est que si d_1 et d_2 sont une paire exacte pour (ce que je vais appeler dans peu de temps les degrés arithmétiques, à savoir) l'idéal I des degrés c tels que $c \leq 0^{(r)}$ pour un certain r (autrement dit c est dans I ssi $c \leq d_1$ et $c \leq d_2$), alors $0^{(\omega)} \leq (d_1 \vee d_2)''$, et c'est le plus grand degré vérifiant cette propriété pour toute paire exacte (d_1, d_2) pour I. Le fait qu'on prenne le second saut de Turing de $d_1 \vee d_2$ ici est pertinent, mais je ne peux pas encore bien expliquer pourquoi à ce stade-là.

En attendant, je remarque que cette construction se relativise bien : on peut définir $d^{(\omega)}$ pour n'importe quel degré d (et ceci constitue une fonction croissante, évidemment). Je ne vais pas essayer de passer à des ordinaux plus grands pour l'instant, je vais en rester à ω.

5 La hiérarchie arithmétique (Σ^0_r, Π^0_r et Δ^0_r)

Je laisse un instant complètement tomber les degrés de Turing. Je veux définir les parties Σ_r (c'est-à-dire Σ^0_r) et Π_r (c'est-à-dire Π^0_r) de $\mathbb{N}$. On pourrait dire que ce sont les parties qui ont une définition en logique du premier ordre qui commence par un quantificateur existentiel (pour Σ) ou universel (pour Π) et qui comporte ensuite $r - 1$ alternations de quantificateurs puis une formule restreinte ; mais comme les gens non logiciens n'aiment globalement pas trop la logique du premier ordre et les

quantificateurs, je vais prendre une définition plus mathématique.

Je pars des parties primitives récursives, ou récursives (ça n'aura pas d'importance) de $\mathbb{N}^k$: je rappelle qu'une partie de $\mathbb{N}^\ell$ avec $\ell < k$ est récursivement énumérable lorsque c'est la projection sur $\mathbb{N}^\ell$ d'une partie primitive récursive, ou simplement récursive, de $\mathbb{N}^k$. J'appellerai « Σ_1 » les parties récursivement énumérables (de $\mathbb{N}^k$), et « Π_1 » leurs complémentaires. Je dis ensuite qu'une partie de $\mathbb{N}^\ell$ avec $l < k$ est « Σ_2 » lorsqu'elle est la projection sur $\mathbb{N}^\ell$ d'une partie Π_1 (=de complémentaire r.é.) de $\mathbb{N}^k$ (avec $k > \ell$, et en fait $k = \ell + 1$ convient), et « Π_2 » lorsqu'elle est le complémentaire d'une partie Σ_2. Et ainsi de suite par récurrence : une partie Σ_{r+1} est la projection d'une partie Π_r qui est elle-même le complémentaire d'une partie Σ_r. Il est évident qu'une partie Σ_r ou Π_r est à la fois Σ_s et Π_s pour tout $s > r$.

Je dirai d'une partie qu'elle est Δ_r lorsqu'elle est à la fois Σ_r et Π_r. Notamment, une partie Δ_1 ce n'est autre qu'une partie (récursivement énumérable et de complémentaire pareil, c'est-à-dire) récursive. Je dirai d'une partie qu'elle est arithmétique lorsqu'elle est Σ_r (ou Π_r) pour un certain r.

Le point-clé qui relie les Σ_r avec les sauts de Turing, c'est le théorème suivant dû à Kleene et Post : une partie de $\mathbb{N}$ est Σ_{r+1} si et seulement si elle est récursivement énumérable relativement à $0^{(r)}$ (le r-ième saut de Turing absolu), et donc Δ_{r+1} ssi elle est récursive relativement à $0^{(r)}$. (Ce n'est pas un résultat très difficile.) Il y a une version relative, aussi : on dit qu'une partie est Σ_r (resp. Π_r, Δ_r, arithmétique) relativement à A lorsqu'elle est vérifie la même définition que les parties Σ_r (resp. Π_r, Δ_r, arithmétiques) mais en partant des ensembles récursifs ou primitifs récursifs relativement à A. Alors le théorème de Kleene et Post affirme qu'une partie est Σ_{r+1} (resp., Δ_{r+1}) relativement à A ssi elle est r.é. (resp., récursive) relativement à $a^{(r)}$ (le r-ième saut de Turing de a).

En particulier, on peut dire d'un degré de Turing qu'il est Δ_r, ou Δ_r relativement à A, lorsqu'il est constitué de parties Δ_r (i.e., qu'il en contient une), ou Δ_r relativement à A : cela revient simplement à dire qu'il est $\leq 0^{(r-1)}$ ou $\leq A^{(r-1)}$. Ceci explique aussi que les degrés de Turing soient parfois appelés les « degrés Δ^0_1 », puisque « être Δ_1 en A » signifie la même chose que « être récursif en A » (par contre, la relation d'être Δ^0_r en A n'est pas transitive si $r \geq 2$). On a aussi la notion de degré arithmétique, plus grossière que celle des degrés de Turing : le degré arithmétique d'une partie de $\mathbb{N}$, c'est l'ensemble de toutes les parties qui sont arithmétiques en celle-ci et réciproquement.

S'agissant des fonctions, on dit qu'une fonction partielle ou totale est Σ_r, ou Δ_r, selon que son graphe l'est. Ceci étant, quand on parle de fonction Σ_r, il y a plus ou moins implicitement l'idée qu'elle est partielle (parce qu'une fonction totale a un graphe Σ_r ssi il est Δ_r), et quand on parle de fonction Δ_r l'idée qu'elle est totale (parce qu'on peut tout aussi bien la rendre totale). J'essaierai de ne pas causer de confusion à ce sujet.

Une autre chose qui est sous-jacente dans ce rapport entre la hiérarchie arithmétique et les degrés de Turing, c'est que pour tout r il existe un ensemble Σ_r universel (on peut prendre, typiquement, l'ensemble énuméré par une machine universelle avec pour

oracle $0^{(r-1)}$ ou, pour r>1 le problème de l'arrêt des machines avec pour oracle $0^{(r-2)}$). C'est-à-dire un ensemble dans Σ_r dans $\mathbb{N}^2$, disons, tel que tout ensemble Σ_r de $\mathbb{N}$ soit une de ses lignes. (Évidemment, en passant au complémentaire, on a un Π_r universel.)

Quant à $0^{(\omega)}$, il peut être représenté par différentes choses : je lui ai donné une définition à base d'arrêts de machines de Turing, mais il est aussi Turing-équivalent à, par exemple, l'ensemble des énoncés vrais dans l'arithmétique du premier ordre. Ça lui donne un certain côté naturel, ça aussi.

6 La hiérarchie analytique (Σ^1_r, Π^1_r et Δ^1_r)

La hiérarchie analytique est quelque chose de beaucoup moins agréable que la hiérarchie arithmétique, et un des thèmes que je veux expliquer c'est qu'elle n'est pas « bonne ». Cette fois il va s'agir de parties définies par des formules du second ordre, c'est-à-dire avec des quantificateurs sur, disons, les fonctions, et cette fois ce que je vais compter c'est le nombre d'alternation de quantificateurs du second ordre (sachant que les quantificateurs du premier ordre ne vont pratiquement plus jouer de rôle). Maintenant les Σ et les Π vont porter un 1 en exposant (au lieu du 0 implicite pour la hiérarchie arithmétique).

Pour définir proprement la hiérarchie analytique, il faut considérer des parties de $\mathbb{N}^k \times (\mathbb{N}^{\mathbb{N}})^\ell$ où $\mathbb{N}^{\mathbb{N}}$ est l'ensemble des fonctions $\mathbb{N} \to \mathbb{N}$. Je commence par considérer les parties récursives de cet espace, « récursif » voulant dire qu'il y a une machine de Turing qui (termine toujours et) calcule la fonction indicatrice de la partie, les entrées étant k entiers naturels et ℓ oracles pour les ℓ fonctions $\mathbb{N} \to \mathbb{N}$. On peut remarquer que, comme une machine de Turing ne peut jamais interroger qu'un nombre fini de valeurs de l'oracle, une partie récursive de $\mathbb{N}^k \times (\mathbb{N}^{\mathbb{N}})^\ell$ est toujours un ouvert fermé (la topologie sur $\mathbb{N}$ étant la topologie discrète et celle sur $\mathbb{N}^{\mathbb{N}}$ la topologie produit).

À partir de ces ensembles, je peux faire deux sortes de projections : les projections de premier ordre (passer de $\mathbb{N}^{k'} \times (\mathbb{N}^{\mathbb{N}})^\ell$ à $\mathbb{N}^k \times (\mathbb{N}^{\mathbb{N}})^\ell$ avec $k' > k$) et celles du second ordre (passer de $\mathbb{N}^k \times (\mathbb{N}^{\mathbb{N}})^{\ell'}$ à $\mathbb{N}^k \times (\mathbb{N}^{\mathbb{N}})^\ell$ avec $\ell' > \ell$), correspondant aux deux sortes de quantifications de la logique du second ordre. J'appelle, comme on s'en doute, « récursivement énumérable » une projection du premier ordre d'un ensemble récursif, « arithmétique » un ensemble qui s'obtient à partir d'un ensemble récursif par n'importe quelle suite (finie) de projections du permier ordre et de passages au complémentaire. J'appellerai aussi $\Sigma^1_0 = \Pi^1_0 = \Delta^1_0$ un ensemble arithmétique. Un ensemble Σ^1_1 c'est une projection du second ordre d'un ensemble arithmétique, et un ensemble Π^1_1 c'est le complémentaire d'un ensemble Σ^1_1.

Il se trouve que les ensembles Σ^1_1 (c'est assez évident) *et* les Π^1_1 (ce n'est pas complètement évident, mais ce n'est pas très dur non plus, grâce à des jeux sur les manipulations de quantificateurs) sont stables par les projections du premier ordre. D'autre part, si on veut, un ensemble Σ^1_1 peut toujours s'écrire comme projection du

second ordre d'un ensemble Π^0_1 (=complémentaire d'un récursivement énumérable) : c'est une façon de chasser presque complètement les quantificateurs du premier ordre.

Ensuite, on continue la hiérarchie de la façon évidente : un Σ^1_2 c'est une projection du second ordre d'un Π^1_1, et un Π^1_2 c'est le complémentaire d'un Σ^1_2, etc. Tous les Σ^1_r et Π^1_r sont stables par projections du premier ordre. Et évidemment j'appelle Δ^1_r un ensemble qui est à la fois Σ^1_r et Π^1_r, et analytique un ensemble qui est Σ^1_r pour un certain r.

Je vais principalement m'intéresser à la hiérarchie analytique sur les parties de $\mathbb{N}$ ou de $\mathbb{N}^k$: j'ai eu besoin de $\mathbb{N}^\mathbb{N}$ dans le cours de la définition pour faire les projections, mais au final il ne m'intéresse plus beaucoup.

On peut facilement définir un ensemble Σ^1_r ou Π^1_r universel : par exemple, un Π^1_1 universel sera donné par l'ensemble des n tels que « pour tout $f : \mathbb{N} \to \mathbb{N}$, la machine de Turing numéro n s'arrête quand on lui passe l'oracle f » (c'est universel parce que j'ai signalé que tout enemble Σ^1_1 était une projection de second ordre d'un co-r.é.), et pour un Σ^1_2 universel, ce sera « il existe $g : \mathbb{N} \to \mathbb{N}$ tel que pour tout $f : \mathbb{N} \to \mathbb{N}$, la machine de Turing numéro n s'arrête quand on lui passe les oracles f et g », etc. Essentiellement, ces Σ^1_r et Π^1_r sont équivalents aux ensembles des formules vraies de l'arithmétique du second ordre ayant le type indiqué. On peut aussi former un exemple d'ensemble qui n'est pas analytique en mettant tout ça ensemble (essentiellement c'est équivalent à l'ensemble des formules vraies de l'arithmétique du second ordre).

Si un ensemble est Δ^1_r, alors tout ensemble qui lui est Turing-équivalent est encore Δ^1_r, donc on peut dire d'un degré de Turing qu'il est Δ^1_r lorsqu'un ou tous les ensembles qu'il contient le sont. (Ceci ne marche pas, évidemment, pour Σ^1_r, mais on peut éventuellement dire d'un degré qu'il est Σ^1_r lorsqu'il contient un ensemble Σ^1_r, comme on l'a fait pour les ensembles récursivement énumérables.)

Mais bon, la hiérarchie analytique n'est pas très intéressante au-delà de Δ^1_2, pour des raisons que j'expliquerai plus loin (mais sommairement, c'est parce qu'elle perd son côté absolu — elle dépend trop de l'univers de la théorie des ensembles — et aussi qu'elle n'admet plus de bonnes hiérarchies intérieures). Donc on va se concentrer sur les ensembles Δ^1_1, Π^1_1 et Δ^1_2. Pour une raison qui apparaîtra progressivement, les ensembles Π^1_1 (et pas les Σ^1_1) vont jouer par rapport aux Δ^1_1 un rôle assez analogues aux ensembles récursivement énumérables (=Σ^0_1) par rapport aux récursifs (=Δ^0_1).

Tout ceci se relativise sans mal : si A est une partie de $\mathbb{N}$, on peut donc définir les ensembles Σ^1_r, Π^1_r, Δ^1_r, analytiques relativement à A (soit en partant des ensembles récursifs relativement à A, soit de façon équivalente en prenant la tranche en A, vu comme élément de $\mathbb{N}^\mathbb{N}$, d'un ensemble Σ^1_r, resp. etc.). Là on voit apparaître une énorme différence avec la hiérarchie arithmétique, c'est que si B est Δ^1_r relativement à A, et que C est Σ^1_r (resp. Π^1_r, resp. Δ^1_r) relativement à B, alors C est Σ^1_r (resp., pareil) relativement à A. En particulier, « être Δ^1_r en » est une relation transitive pour chaque r. Ce n'est pas très surprenant parce qu'on permet justement des quantifications sur les parties, mais évidemment « être Δ^0_r en » n'était pas transitive lorsque $r \geq 2$.

Les ensembles Δ^1_1 (resp., Δ^1_1 relativement à A) s'appellent « hyperarithmétiques » (resp., hyperarithmétiques relativement à A). Les classes d'équivalences pour la relation « être Δ^1_1 en, et réciproquement » s'appellent les hyperdegrés. Maintenant je vais essayer d'expliquer pourquoi les ensembles hyperarithmétiques sont quelque chose de très naturel (beaucoup plus que les ensembles arithmétiques). Par ailleurs, je dirai d'une fonction *totale* qu'elle est hyperarithmétique (ou plus généralement, Δ^1_r) lorsque son graphe l'est ; et d'une fonction *partielle* qu'elle est Π^1_1 lorsque son graphe l'est (les raisons de ce choix précis apparaîtront plus tard).

7 Il existe des ensembles hyperarithmétiques non arithmétiques

J'ai défini les ensembles hyperarithmétiques comme ceux qui sont Δ^1_1 dans la hiérarchie analytique (i.e., à la fois projection selon $\mathbb{N}^\mathbb{N}$ du complémentaire d'un récursivement énumérable, et complémentaire de la projection selon $\mathbb{N}^\mathbb{N}$ d'un ensemble récursivement énumérable). Vu comme ça, ce n'est pas une notion très jolie. Si mon but est de convaincre qu'elle est naturelle, je vais en donner plusieurs autres caractérisations.

Mais d'abord il faut que je donne un exemple d'ensemble (ou de degré) hyperarithmétique qui ne soit pas arithmétique : c'est $0^{(\omega)}$. Ce n'est pas évident de voir pourquoi il est hyperarithmétique, alors je vais essayer d'esquisser la raison : le truc est que la fonction $H(r,n)$, qui vaut 1 ou 0 selon que la n-ième machine de Turing disposant de l'oracle $H(r-1,\bullet)$ s'arrête ou pas, est arithmétiquement implicitement définissable, ce qui signifie qu'on peut écrire une formule du premier ordre avec une variable libre F du second ordre qui définit « $F = H$ », ou, plus concrètement, qu'il existe une « machine arithmétique » qui va renvoyer « oui » ou « non » selon qu'un oracle F qu'on lui a passé est ou non cette fameuse fonction H ; et par « machine arithmétique » je veux dire une machine de Turing disposant d'un nombre fini (en l'occurrence, 2) de niveaux d'oracles d'arrêt (mais d'oracles d'arrêt *relativement à l'oracle F fourni en paramètre*, évidemment !). Or ça, ce n'est pas très difficile à voir : il s'agit essentiellement de tester que $H(r,n)$ vaut bien ce qu'il est censé valoir, et comme on a assez de niveaux d'oracles d'arrêt (ou, côté logique, assez de niveaux de quantificateurs du premier ordre) on peut faire des vérifications infinies — il suffit d'écrire la définition de l'oracle de l'arrêt. Ceci prouve bien que H est Δ^1_1 (i.e., que son graphe l'est), en l'écrivant soit sous la forme « il existe un F tel que $\langle$ test arithmétique de $F = H$ $\rangle$ qui prend cette valeur » soit « pour tout F tel que $\langle$ test arithmétique de $F = H$ $\rangle$, F prend cette valeur ».

Ceci montre que « hyperarithmétique » est strictement plus gros que « arithmétique », mais devrait aussi convaincre que les fonctions hyperarithmétiques sont assez puissantes : je viens d'expliquer, là, que la fonction qui renvoie « vrai » ou « faux » quand on lui présente un énoncé arithmétique du premier ordre, est hyperarithmétique

(puisqu'elle a le degré $0^{(\omega)}$). Et en fait, on peut plus ou moins se douter, en regardant le procédé de la démonstration que je viens de donner, que les degrés hyperarithmétiques vont contenir non seulement 0, $0'$, $0''$, ..., $0^{(\omega)}$, $0^{(\omega+1)}$, ..., mais aussi toute itération du saut de Turing selon un ordinal « calculable » dans un certain sens par une machine de Turing.

8 Fonctions calculables avec l'hyperoracle du saut ou l'hyperoracle E

La terminologie « hyperoracle » est de moi (le terme standard est « (fonction récursive relativement à) une fonctionnelle de type 2 », qui est long et moche), mais je pense qu'elle est assez naturelle.

Une machine de Turing avec oracle, c'est une machine qui peut interroger à tout moment une fonction (totale) $O : \mathbb{N} \to \mathbb{N}$. Une machine de Turing avec hyperoracle, c'est une machine qui peut interroger une fonction totale $O : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ (je veux dire, une fonction de $\mathbb{N}^{\mathbb{N}}$ vers $\mathbb{N}$). Ceci mérite des explications, bien sûr : comment est-on censé interroger une fonction dont l'argument est dans $\mathbb{N}^{\mathbb{N}}$? Il faut lui fournir une fonction $\mathbb{N} \to \mathbb{N}$ et celle-ci sera, évidemment, calculée par la même machine (avec hyperoracle), mais en un nombre infini d'étapes. Plus exactement, quand la machine de Turing décide de faire appel à l'hyperoracle, elle lance en parallèle une infinité de calculs $f(0)$, $f(1)$, $f(2)$, $f(3)$, etc., pour f une fonction donnée par un programme dans le même langage (et qui a lui-même le droit de faire appel à l'hyperoracle) : si ne serait-ce qu'un seul des calculs $f(k)$ ne termine pas, alors la machine appelante ne termine pas ; en revanche, si toutes terminent, alors la machine reçoit la valeur $O(f)$ retournée par l'hyperoracle. Dit comme ça, la notion a l'air un peu foireuse, mais si on y réfléchit on devrait pouvoir se convaincre que c'est bien défini et arriver à une caractérisation formelle.

Il faut se méfier des hyperoracles, parce que l'intuition qu'on peut avoir des machines de Turing est un peu cassée en leur présence. Le principal problème est qu'alors que pour les machines de Turing (avec oracle) normales on peut trouver une fonction (primitive) récursive avec l'oracle qui énumère les N premières étapes du calcul, et dire quelque chose comme « la machine s'arrête ss'il existe un N tel que les N premières étapes du calcul se terminent par "j'ai fini" », ceci ne marche plus avec un hyperoracle. Néanmoins, pas mal de choses sont vraies quand même (par exemple, si une fonction $H : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ est calculable à partir d'une autre par une machine avec hyperoracle G + un oracle pour lui passer l'argument $\mathbb{N} \to \mathbb{N}$, et qu'une autre fonction G est calculable à partir de F, alors H est calculable à partir de F).

Le principal hyperoracle qui va m'intéresser, c'est l'hyperoracle E : la fonction $\mathsf{E} : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ est donnée par $\mathsf{E}(f) = 1$ si 0 est dans l'image de la fonction (totale)

f, et $\mathsf{E}(f) = 0$ sinon. On aura deviné que les valeurs 0 et 1 n'ont aucune importance : les machines de Turing avec l'hyperoracle E, ce sont des machines qui à tout moment peuvent calculer une infinité de valeurs $f(0)$, $f(1)$, $f(2)$,... (la fonction f étant elle-même calculée par la même machine et pouvant faire appel à l'hyperoracle), et si tous ces calculs retournent, la machine a le pouvoir de tester si, par exemple, l'une d'entre elle a renvoyé telle ou telle valeur (par contre, si l'un des calculs $f(i)$ ne termine pas, la machine appelante ne termine pas non plus).

Évidemment, une machine avec hyperoracle E peut décider l'arrêt des machines de Turing (facile : on prend la fonction $N \mapsto f(N)$ qui regarde si la machine considérée s'arrête en N étapes et renvoie 0 si oui, 42 si non), ou des machines de Turing avec oracle de l'arrêt des machines de Turing, ou des machines de Turing avec cet oracle-là, etc., et même on se convainc facilement qu'elles peuvent calculer la fonction $H(r, n)$ qui décide si la n-ième machine de Turing disposant de l'oracle $H(r-1, \bullet)$ s'arrête ou pas (i.e., une machine avec hyperoracle E peut décider si un énoncé arithmétique du premier ordre est vrai ou pas). En fait, on peut se convaincre que l'hyperoracle E est équivalent à l'hyperoracle donné par le saut de Turing lui-même (i.e., la fonction $\mathrm{TJ} : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ donnée par $\mathrm{TJ}(f)$ = les numéros des machines de Turing qui s'arrêtent quand elles disposent de f comme oracle) — équivalent au sens où toute fonction calculable avec l'un est calculable avec l'autre (ou, en fait, que chacun des hyperoracles TJ et E est calculable à partir de l'autre dans le sens que j'ai esquissé plus haut).

J'appellerai « machine hyperarithmétique » une machine de Turing disposant de E, ou de TJ, pour hyperoracle.

Et la caractérisation qu'on aura sans doute devinée d'après ce nom, c'est : une partie de $\mathbb{N}^k$ (resp., une fonction totale $\mathbb{N}^k \to \mathbb{N}$) est hyperarithmétique=$\Delta^1_1$ ssi sa fonction indicatrice (resp., la fonction elle-même) est calculable par une machine hyperarithmétique (et terminant toujours, bien sûr). S'agissant des fonctions partielles, le résultat est : une partie de $\mathbb{N}^k$ est Π^1_1 (resp., une fonction partielle $\mathbb{N}^k \to \mathbb{N}$ a un graphe Π^1_1) ssi sa fonction indicatrice est l'ensemble des entrées sur lesquelles une machine de Turing hyperarithmétique termine (resp., si la fonction elle-même est calculable par une telle machine).

Une différence essentielle avec les machines de Turing usuelles est la suivante : je viens de dire que tout ensemble Π^1_1 de $\mathbb{N}$ peut s'écrire comme l'ensemble des valeurs sur lesquelles où une certaine machine hyperarithmétique termine (et réciproquement) ; l'analogie avec les machines de Turing suggérerait d'appeler ça « hyperarithmétiquement énumérable » et de penser que ce soit aussi l'image d'une fonction totale hyperarithmétique — or *ce n'est pas vrai* : l'image d'une fonction hyperarithmétique totale $\mathbb{N} \to \mathbb{N}$ est un ensemble hyperarithmétique, et c'est même évident puisqu'une machine hyperarithmétique peut savoir si une fonction hyperarithmétique va finir par prendre telle ou telle valeur donnée ! (Du coup, on

ne peut que souligner de nouveau qu'il n'est pas possible d'« énumérer les étapes » d'une machine hyperarithmétique comme on peut le faire pour une machine de Turing, sinon les machines hyperarithmétiques décideraient leur propre arrêt, ce qui n'est pas le cas.) Par contre, un ensemble Π^1_1 peut toujours s'écrire comme l'ensemble des valeurs énumérées par une machine hyperarithmétique si celle-ci n'est pas obligée de toujours s'arrêter (ça définit une fonction partielle Π^1_1), donc la terminologie « hyperarithmétiquement énumérable » pour les Π^1_1 se défend quand même si on fait très attention et qu'on choisit d'appeler « fonction hyperarithmétique partielle » une fonction dont le graphe est Π^1_1.

Une bonne nouvelle, c'est que tout ceci se relativise bien : on peut définir sans mal les machines hyperarithmétiques avec un oracle, et il est encore vrai que la fonction indicatrice de B est calculée par une machine hyperarithmétique ayant A pour oracle (resp., B est l'ensemble de définition d'une fonction partielle calculée par une telle machine) ssi B est Δ^1_1(=hyperarithmétique) relativement à A (resp., B est Π^1_1 relativement à A).

9 Notations ordinales de Kleene

Pour arriver à donner une nouvelle caractérisation des fonctions hyperarithmétiques, il faut que je parle un peu des notations ordinales à la Kleene.

Je rappelle d'abord qu'un ordinal, c'est une classe d'équivalence d'ensembles bien ordonnés, un ensemble étant dit « bien ordonné » lorsque toute partie non vide a un plus petit élément (ou, de façon plus visuelle, lorsqu'il est totalement ordonné et n'admet pas de suite infinie strictement décroissante ; ou, mieux, un ensemble totalement ordonné tel que lorsqu'une partie de l'ensemble contient un élément dès qu'elle contient tous les éléments strictement plus petits, alors cette partie est l'ensemble tout entier) ; de deux ensembles bien ordonnés il y en a toujours un qui est isomorphe à un segment initial de l'autre ou à l'autre tout entier ; et l'isomorphisme est alors unique. Les ordinaux sont eux-mêmes bien ordonnés dans le sens évident. Chaque ordinal a un successeur (qui se fabrique en ajoutant un dernier élément à l'ensemble), et chaque ensemble d'ordinaux a une borne supérieure qui est un ordinal (en particulier, chaque suite croissante d'ordinaux a une borne supérieure qu'on appellera la limite de cette suite). Tout ordinal est soit 0 (l'ordinal de l'ensemble vide), soit le successeur d'un unique autre ordinal, soit un ordinal limite c'est-à-dire la borne supérieure des ordinaux strictement plus petit ; et si l'ordinal est dénombrable, ce dernier cas se dit simplement : c'est la limite d'une suite strictement croissante d'ordinaux plus petits.

Une notation ordinale à la Kleene est soit (a) le nombre 1, qui désigne l'ordinal 0, soit (b) le nombre 2^n où n est une notation ordinale de Kleene, qui désigne alors l'ordinal successeur de l'ordinal désigné par n, soit (c) le nombre $3 \cdot 5^n$ où n est le numéro d'une machine de Turing qui énumère une suite de notations ordinales désignant des ordinaux

strictement croissants, auquel cas ceci désigne la borne supérieure de ces ordinaux. (Si on veut un truc formellement bien défini, il faut définir à la fois la notion de « notation ordinale de Kleene » et celle de « ordinal (dénombrable) désigné par une telle notation », comme les plus petits tels que gnagnagna. Évidemment, les écritures 2^n et $3 \cdot 5^n$ servent juste à coder les trucs en question.) Il est essentiel de se rendre compte qu'un même ordinal (n'importe quel ordinal infini) peut très bien admettre plein de notations, et que rien ne permet de savoir si un entier naturel donné est ou n'est pas une notation ordinale (i.e., je n'affirme certainement pas que ce soit décidable, et on va voir que c'est, justement, un problème très dur).

On définit une relation binaire $<_O$ sur les entiers naturels par la clôture transitive de : $n <_O 2^n$ pour tout n, et $m <_O 3 \cdot 5^n$ pour tout n et tout m énuméré par la n-ième machine de Turing. La restriction de $<_O$ à l'ensemble O des entiers qui sont vraiment des notations ordinales définit un ordre partiel (en fait, une structure d'arbre infini) pour lequel si $m <_O n$ alors les ordinaux désignés par m et n sont dans l'ordre attendu ; et mieux : si n est dans O alors tout m tel que $m <_O n$ est dans O et l'ensemble de ces m est bien ordonné par $<_O$ avec pour type l'ordinal désigné par n. Sur les entiers naturels en général, cette relation n'a aucune propriété sympa (elle n'est même pas irréflexive), mais elle a quand même le bon goût d'être récursivement énumérable, ce qui est utile pour prouver beaucoup de choses sur O (en gros, *la* question importante, c'est de décider si un élément est dans O, tout le reste en découle). Dans le même genre d'idée, on peut définir une fonction récursive partielle sur les entiers qui est définie sur tout couple d'éléments de O et représente la somme, le produit ou l'exponentielle des ordinaux représentés par les arguments.

Un ordinal est dit constructif ssi il admet une notation à la Kleene (i.e., s'il est représenté par un élément de O), et il est dit récursif ssi c'est le type d'une certaine relation de bon ordre récursive sur les entiers naturels. En fait, ces deux notions coïncident : si un ordinal est constructif et si n en est une notation, on voit facilement qu'en énumérant $\{m | m <_O n\}$ par les entiers naturels on obtient une relation de bon ordre sur ceux-ci qui a pour type l'ordinal désigné par n ; et pour la réciproque, le mieux est de montrer que si un ordinal α est récursif alors $\omega \cdot \alpha$ a une notation (l'intérêt du ω c'est qu'a priori on ne peut pas tester si α est successeur, mais en fait on s'en fout), et du coup α l'est.

En particulier, le plus petit ordinal qui n'est pas le type d'une relation de bon ordre récursif sur $\mathbb{N}$ coïncide avec le plus petit qui n'a pas de notation à la Kleene : cet ordinal s'appelle le ω_1 de Church-Kleene (ou le « ω_1 constructif » — terminologie vraiment pourrie parce qu'il y a aussi un « ω_1 constructible » qui est beaucoup plus grand — ou, chez certains auteurs, juste ω_1, ce qui est franchement abusif ; on l'appelle aussi « ω_1 admissible » pour des raisons qui apparaîtront plus loin).

10 Itération du saut de Turing le long de notations ordinales, hiérarchie hyperarithmétique

Je peux maintenant définir le saut de Turing itéré le long d'une notation ordinale : je définis une fonction $H(o, \bullet)$, pour o dans l'ensemble O des notations ordinales, de la façon suivante : (a) j'appelle $H(1, \bullet)$ la fonction constante nulle (je rappelle que 1 désigne l'ordinal 0, c'est une convention à la con), (b) j'appelle $H(2^n, \bullet)$ la fonction d'arrêt des machines de Turing avec pour oracle $H(n, \bullet)$ (je rappelle que 2^n désigne l'ordinal successeur de celui désigné par n) et (c) j'appelle $H(3 \cdot 5^n, \bullet)$ la fonction qui à un couple $\langle r, k \rangle$ associe $H(m, k)$ où m est le r-ième nombre énuméré par la n-ième machine de Turing. Il se trouve alors que le degré de Turing de $H(o, \bullet)$, pour o une notation ordinale, ne dépend que de l'ordinal α désigné par celle-ci : on l'appelle α-ième saut de Turing absolu et on le note $0^{(\alpha)}$. Il est assez clair que les premiers sauts de Turing ainsi définis (pour $\alpha \leq \omega$, disons) sont les mêmes que ceux qu'on avait déjà définis.

Le résultat important est alors : un ensemble est hyperarithmétique ssi sa fonction indicatrice est récursive en l'un des $H(o, \bullet)$ pour o une notation ordinale (autrement dit, récursive dans le α-ième saut de Turing absolu, pour α un ordinal récursif). De même, une fonction *totale* est hyperarithmétique ssi elle est récursive dans l'un des $H(o, \bullet)$ pour o une notation ordinale.

Donc non seulement on a une nouvelle caractérisation des ensembles hyperarithmétiques, mais en plus on a une hiérarchie à l'intérieur de Δ^1_1 indicée par les ordinaux inférieurs au ω_1 de Church-Kleene : pour un $\alpha < \omega_1^{\mathrm{CK}}$, on définit les ensembles $\Sigma_{1+\alpha}$ (ou $\Sigma^0_{1+\alpha}$, ou $\Sigma^{\mathrm{Hyp}}_{1+\alpha}$ si on veut être bien précis) comme ceux qui sont récursivement énumérables sous l'oracle $H(o, \bullet)$ avec o n'importe quelle notation désignant α ; et évidemment, on définit les ensembles $\Pi_{1+\alpha}$ comme les complémentaires des ensembles $\Sigma_{1+\alpha}$, et les $\Delta_{1+\alpha}$ comme ceux qui sont à la fois $\Sigma_{1+\alpha}$ et $\Pi_{1+\alpha}$ (autrement dit, les ensembles qui ont un degré de Turing inférieur ou égal à $0^{(\alpha)}$). Les ensembles hyperarithmétiques sont ceux qui sont $\Delta^0_{1+\alpha}$ pour un certain α. (J'ai décalé les indices finis de 1 pour coïncider avec les notations déjà données pour la hiérarchie arithmétique : mais si α est infini alors $1 + \alpha$ égale α.)

Accessoirement, il existe une fonction *récursive* (totale) qui, donné le numéro d'une machine hyperarithmétique qui termine toujours, fournit une notation ordinale o et le numéro d'une machine de Turing telle que cette dernière avec oracle $H(o, \bullet)$ fasse le même travail que la machine hyperarithmétique donnée (c'est assez facile, en fait, parce qu'on cache dans o tout le boulot fait par la machine hyperarithmétique). Évidemment, la fonction récursive ne peut pas décider si la machine hyperarithmétique termine : on voit donc de nouveau que décider si un nombre est une notation ordinale est un problème très dur (non hyperarithmétique) — je vais revenir là-dessus.

Une autre chose sous-jacente dans ces résultats, c'est que ω_1^{CK}, qui est défini comme

le plus petit ordinal qui ne peut pas s'écrire comme le type d'un bon ordre *récursif* sur les entiers, est aussi le plus petit ordinal qui ne peut pas s'écrire comme le type d'un bon ordre *hyperarithmétique* sur les entiers.

Tout ceci se relativise, *mais* il faut faire attention au fait que, si l'oracle A n'est pas hyperarithmétique=Δ^1_1, l'ordinal $\omega_1^{\mathrm{CK}}[A]$ de Church-Kleene relatif à cet oracle A peut être plus grand que le ω_1^{CK} absolu. Si on veut retrouver la notion d'ensemble hyperarithmétique relativement à A, il faut naturellement pousser la hiérarchie hyperarithmétique relativisé à A jusqu'à ce $\omega_1^{\mathrm{CK}}[A]$. Au passage, on a le résultat suivant : si B est $\Delta_{1+\alpha}$ relativement à A et que C est $\Delta_{1+\beta}$ relativement à B, alors C est $\Delta_{1+\alpha+\beta}$ relativement à A (attention, l'addition sur les ordinaux n'est pas commutative : un ensemble Δ_ω en un ensemble Δ_2 est Δ_ω alors qu'un ensemble Δ_2 en un ensemble Δ_ω est $\Delta_{\omega+1}$).

11 L'hypersaut

Je rappelle que j'ai défini les hyperdegrés comme les classes d'équivalence pour la relation « machin est hyperarithmétique relativement à truc, et réciproquement », avec la relation d'ordre partiel « être hyperarithmétique en ». Il y a une certaine analogie avec les degrés de Turing ; mais cette analogie est trompeuse, car les hyperdegrés sont plutôt moins compliqués que les degrés de Turing.

D'abord, on peut donner un exemple d'ensemble qui soit Π^1_1 mais pas hyperarithmétique : l'ensemble O des notations ordinales à la Kleene. Si on n'aime pas les notations ordinales, on peut donner plusieurs ensembles qui sont équivalents (au moins au sens de Turing, et même dans des sens plus fins comme la réduction many-to-one) à O. Par exemple, l'ensemble des machines de Turing qui définissent un bon ordre sur les entiers naturels. Ou bien, l'ensemble des numéros des ensembles récursifs de suites finies d'entiers qui sont un arbre (=stables par racourcissement de la suite) n'ayant aucune branche infinie (c'est un peu plus compliqué à dire, mais c'est en fait plus commode à manier). Ou, tout simplement, l'ensemble des machines hyperarithmétiques qui ne terminent pas (je dis « ne terminent pas » plutôt que « termine » pour avoir un ensemble Π^1_1 comme les précédents). Il n'est pas très difficile de se convaincre que ces ensembles sont Π^1_1 (par exemple, dire qu'un truc est un bon ordre se dit « il n'existe pas de suite infinie strictement décroissante pour ce bon ordre », ce qui est bien Π^1_1), et même qu'ils sont complets/universels, au moins au sens où tout ensemble Π^1_1 est Turing-réductible à eux (on a même mieux, réductibles many-to-one, mais peu importe) : savoir décider l'appartenance à un de ces ensembles permet de décider l'arrêt des machines hyperarithmétiques. Là, ça ressemble très fort au problème de l'arrêt, et, de fait, le degré de Turing (et a fortiori l'hyperdegré) d'un quelconque de ces ensembles va s'appeler le premier hypersaut (absolu).

Là où il y a une grosse différence avec le saut de Turing, c'est qu'en fait il n'y a

pas d'analogue au « problème de Post » : *tout* ensemble Π^1_1 (=énuméré par une machine hyperarithmétique qui peut ne pas s'arrêter) A qui n'est pas Δ^1_1 (=hyperarithmétique) a l'hyperdegré de O (ou d'un quelconque des ensembles décrits ci-dessus). Pour ça, il faut d'abord voir que si A est Π^1_1, alors il existe une fonction f récursive telle que x soit dans A ssi $f(x)$ est dans O (on dit que tout ensemble Π^1_1 est réductible à O au sens de la réduction many-to-one, qui est plus fine que celle de Turing) : ça ce n'est pas très difficile, ça fait partie de l'universalité de O (et d'ailleurs la même chose est vraie *mutatis mutandis* pour les ensembles récursivement énumérables et le problème de l'arrêt). Or les ordinaux désignés par $f(x)$ ne peuvent pas être bornés sous ω_1^{CK}, car il n'est pas très difficile de se convaincre que l'ensemble des notations ordinales désignant un ordinal borné par une certaine borne est hyperarithmétique (=Δ^1_1) : mais du coup, donné un oracle qui énumère A, on peut énumérer O par une machine hyperarithmétique en cet oracle et qui termine toujours (en énumérant pour chaque x dans A l'ensemble des notations ordinales plus petites que $f(x)$) ; et comme pour des machines hyperarithmétiques s'arrêtant toujours, énumérer ou décider c'est la même chose, on en déduit que O est Δ^1_1 en A (et tout ensemble Π^1_1 est Δ^1_1 en A).

Tout ceci se relativise très bien, et si A est une partie quelconque de $\mathbb{N}$ (ou une fonction totale $\mathbb{N} \to \mathbb{N}$), j'appelle hypersaut $\mathrm{HJ}(A)$ de A le degré de Turing (et a fortiori l'hyperdegré plus grossier que lui) donné par l'ensemble $O[A]$ des notations ordinales définies par des machines ayant A pour oracle — on peut préférer voir ça comme le problème de l'arrêt pour les machines hyperarithmétiques ayant A comme oracle. On a un ordinal naturellement associé à la situation (le sup des ordinaux dénotés par les éléments de $O[A]$) qui est $\omega_1^{\mathrm{CK}}[A]$.

Pour les ordinaux associés aux hypersauts successifs, on utilise une notation un peu plus intelligente : on appelle ω_2^{CK}, ω_3^{CK}, et ainsi de suite, les ordinaux associés aux hypersauts absolus. En fait, il n'y a pas de subtilité particulière : si on considère l'ensemble des ordinaux dénombrables qui sont soit « le plus petit ordinal qui n'est pas le type d'une relation de bon ordre sur $\mathbb{N}$ récursive relativement à l'oracle A » (pour un certain A), aka $\omega_1^{\mathrm{CK}}[A]$, *ou* limite de tels ordinaux, et qu'on appelle $\omega_\alpha^{\mathrm{CK}}$ le α-ième tel ordinal, alors les ordinaux des premiers hypersauts sont bien ω_2^{CK}, ω_3^{CK}, et ainsi de suite au moins sur les entiers naturels. Attention quand même, ici j'ai défini $\omega_\omega^{\mathrm{CK}}$ comme la limite des ω_i^{CK} avec i entier naturel, tous les auteurs ne sont pas forcément d'accord avec ça (notamment, mon $\omega_\omega^{\mathrm{CK}}$ n'est pas « admissible » dans un sens que je n'ai pas défini, il est seulement limite d'admissibles).

12 Quelques mots sur les machines ordinales

Il existe une généralisation des fonctions récursives pour travailler sur les ordinaux au lieu de travailler sur les entiers naturels. Je n'ai pas le courage de la définir formellement dans ce griboullis (ce n'est pas spécialement difficile, c'est juste un peu

long à faire parfaitement). J'en donne juste une description très sommaire.

On fixe un ordinal limite δ (imaginez typiquement ω_1^{CK}), et on considère un langage de programmation dans lequel les variables prennent des valeurs qui sont des ordinaux, on a les opérations arithmétiques usuelles (et la comparaison, et un moyen standard de créer des paires) sur ces ordinaux, et on a aussi une boucle « grand while » (ou « while ordinal », que sais-je) qui effectue un nombre d'itérations donné par un ordinal : aux temps ordinaux successeurs, il se passe la même chose qu'avec une boucle while normal, mais on peut aussi franchir les ordinaux limites, et alors les variables de la boucle prennent une valeur qui est la limite sup des valeurs antérieures de cette variable. On peut soit exécuter une boucle de ce type jusqu'à un ordinal ρ reçu en argument ou préalablement calculé, ces boucles terminant toujours (si les étapes de la boucle terminent elles-mêmes, bien sûr) ; soit exécuter une boucle de taille δ : si la condition de sortie de boucle n'est jamais réalisée en un temps $< \delta$, le programme ne termine pas ; si elle est réalisée, la boucle renvoie ce qu'elle veut renvoyer.

[Addendum[

Pour que ça soit parfaitement clair, il y a deux sortes de boucles qui sont permises à une δ-machine (j'avais oublié la première dans une précédente version de ce griboullis) :

– les boucles bornées par un argument ρ spécifié préalablement à l'entrée de la boucle : ces boucles ont un temps qui parcourt les ordinaux jusqu'à ρ exclu (ou jusqu'à une condition de sortie optionnelle), et si tous ces ordinaux sont traversés ce n'est pas un problème (on fait comme si on était arrivé au temps ρ, et on quitte la boucle immédiatement), ces boucles terminent toujours si toutes les étapes terminent ; elles garantissent qu'une δ-machine peut toujours émuler une γ-machine si on lui passe γ en argument ; et

– les boucles non bornées : ces boucles ont un temps qui parcourt les ordinaux jusqu'à δ exclu (l'ordinal δ étant spécifié avec la machine) ou jusqu'à une condition de sortie indispensable, et si tous les ordinaux $< \delta$ sont traversés alors la machine ne termine pas (il est donc indispensable, si on veut terminer, de rencontrer la condition de sortie).

Dans les deux cas, on franchit les temps ordinaux limites en prenant la limite sup sur les valeurs des variables. Si on se place dans le cas où δ est un ordinal admissible (cf. plus bas) et qu'on ne considère que des arguments $< \delta$, alors on peut effectivement oublier la première sorte de boucle (elle est évidemment émulable avec la seconde en mettant « temps$\leq \rho$ » comme condition de sortie). Par contre, si on prend $\delta = \omega$ (la plus petite valeur autorisée), les boucles de la première sorte sont beaucoup plus puissantes ; si je ne me trompe pas, les premiers ordinaux ω-stables sont les mêmes que les premiers ordinaux admissibles (i.e., eux-mêmes-stables), soit ω_1^{CK}, ω_2^{CK}, ω_3^{CK}, etc. : le premier qui ne soit pas admissible est leur limite $\omega_\omega^{\mathrm{CK}}$.

]]

(Pinailliage sur la récursion : le plus simple est de l'interdire, c'est-à-dire d'imposer qu'une fonction ne peut appeler qu'une fonction définie strictement plus haut, comme on fait pour le langage définissant les fonctions p.r. ; de toute façon, le programmeur peut émuler la récursion en simulant lui-même une pile grâce à des suites finies d'ordinaux codées comme des paires. Si on autorise la récursion, le plus simple est de décider qu'une récursion non bornée cause un programme qui ne termine pas — si on veut traverser des temps ordinaux, il faudra obligatoirement passer par un grand while.)

On appelle fonction δ-récursive (sans paramètres) sur les ordinaux, ou sur un ordinal λ, une fonction calculée par un tel programme (il y a un certain nombre de variations autour de cette définition, sans grande conséquence sur ce que je veux en dire). On appelle fonction δ-récursive *en paramètres* sur les ordinaux, ou sur un ordinal λ, une fonction calculée par un tel programme à partir de certaines constantes $< \lambda$ (c'est-à-dire, une application partielle d'une fonction δ-récursive à des ordinaux $< \lambda$) : cela fait une différence parce que les constantes $< \lambda$ ne sont pas nécessairement δ-récursives pour tout δ et tout λ.

Évidemment, si on augmente la valeur de δ, des programmes qui ne terminaient pas peuvent se mettre à terminer : donc il faut bien préciser pour quel δ on parle (et parfois il faut préciser sur que λ on vit ou dans quel λ on admet des paramètres). Pour $\delta = \omega$, le grand while est exactement le while usuel, et sur les entiers naturels on obtient un truc équivalent à Turing ; comme les seules constantes ordinales qu'on peut écrire dans le programme sont des entiers naturels, pour $\delta = \omega$, on n'a pas moyen de fabriquer des ordinaux infinis si on n'en reçoit pas en entrée, et la fonction prend des valeurs qui sont des entiers naturels sur les entiers naturels (sur lesquels elle est définie). Ceci illustre le fait qu'une fonction constante n'est pas forcément δ-récursive : la fonction constante ω n'est pas ω-récursive (sans paramètres ; par contre, elle est trivialement ω-récursive si on autorise des paramètres $\geq \omega$).

Si un ordinal $\delta > \omega$ est tel que toute fonction δ-récursive, quand on lui passe des valeurs $< \delta$, et qu'elle termine, renvoie toujours un résultat $< \delta$, alors on dit que δ est un ordinal « admissible », ou « récursivement régulier ». Autrement dit, dire que δ est admissible signifie qu'avec des ordinaux $< \delta$ on ne peut fabriquer que des ordinaux $< \delta$ en utilisant de la δ-récursion (des grands while de longueur δ). Savoir si ω est admissible est une question de définition, mieux vaut éviter de se la poser.

Le plus petit ordinal admissible $> \omega$ est ω_1^{CK}. De plus : une partie de ω, ou une fonction totale $\omega \to \omega$, est ω_1^{CK}-récursive si et seulement si elle est hyperarithmétique.

Ceci se relativise par rapport à un oracle A, mais il faut faire attention : une partie de ω, ou une fonction totale $\omega \to \omega$, est $\omega_1^{\mathrm{CK}}[A]$-récursive relativement à A si et seulement si elle est hyperarithmétique relativement à A. Le fait que ce soit $\omega_1^{\mathrm{CK}}[A]$ et pas ω_1^{CK} qui intervienne est critique : ceci explique que la théorie de la ω_1^{CK}-calculabilité et la théorie des hyperdegrés ne sont pas la même chose (j'ai expliqué qu'il n'y avait pas de

problème de Post pour les hyperdegrés : en revanche, il y en a bien un, et il est assez subtil, pour les degrés de Turing sur ω_1^{CK}).

13 Quelques mots sur la hiérarchie constructible

La hiérarchie L_α de Gödel (avec α un ordinal) est définie par : L_0 = l'ensemble vide, ou bien l'ensemble des ensembles hériditairement finis, $L_{\alpha+1}$ = l'ensemble de toutes les parties de L_α définies par une formule du premier ordre dans le langage de la théorie des ensembles (en autorisant les éléments de L_α comme constantes), et pour δ limite, L_δ = la réunion des L_α pour $\alpha < \delta$. Il y a d'autres définitions possibles si on n'aime pas celle-là (par exemple, $L_{\alpha+1}$ peut se définir comme l'intersection entre l'ensemble des parties de L_α et la « clôture rudimentaire » de $L_\alpha \cup \{L_\alpha\}$, c'est-à-dire sa clôture sous l'effet d'un certain nombre d'opération dites opérations de Gödel). On appelle L la classe réunion de tous les L_α : l'axiome $V = L$ affirme que tout ensemble est dans L, et il n'est ni démontrable ni réfutable dans ZFC (il implique l'hypothèse généralisée du continu et, dans ZF, il implique l'axiome du choix). Par ailleurs, il y a des résultats du type absoluïté des concepts, je ne veux pas trop rentrer dans les détails (du genre, $V = L$ est vrai dans L, parce que les L_α sont absolus).

Le rapport entre les L_α et la théorie de la calculabilité supérieure, c'est que : (1) un ordinal $\delta > \omega$ est admissible (au sens défini ci-dessus, i.e., stable par les fonctions δ-récursives) ssi L_δ est un modèle d'un fragment raisonnable de la théorie des ensembles, à savoir « Kripke-Platek » (c'est ZFC sans l'axiome de l'ensemble des parties, avec l'axiome de séparation limité aux formules à quantificateurs bornés, et l'axiome de remplacement remplacé par l'axiome de collection lui aussi limité à de telles formules), et (2) lorsque c'est le cas, une fonction définie sur un ordinal $< \delta$ est δ-récursive ssi elle appartient à L_δ (et une fonction définie sur δ est δ-récursive ssi elle est Δ_1-définissable sur L_δ).

En particulier, les parties de ω (ou fonctions $\omega \to \omega$) qui sont dans $L_{\omega_1^{\mathrm{CK}}}$ sont exactement les parties (ou fonctions) hyperarithmétiques, ce qui fournit une nouvelle équivalence dans mon puzzle.

On peut par ailleurs définir une hiérarchie qui raffine la hiérarchie constructible : j'appelle $M_{\omega\alpha} = L_\alpha$ (ici je fais la convention que L_0 est l'ensemble des ensembles hériditairement finis), et $M_{\omega\alpha+n}$ = l'ensemble de toutes les parties de L_α définies de façon équivalente par deux formules du premier ordre dans le langage de la théorie des ensembles qui ont n alternations de quantificateurs devant une formule à quantificateurs bornés, l'une commençant par des existentiels et l'autre par des universels (bref, les parties Δ_n de L_α). Ainsi, une partie de ω (ou une fonction $\omega \to \omega$) est dans M_1 ssi elle est récursive(=Δ_1), elle est dans M_2 ssi elle est Δ_2, etc., et la hiérarchie M_α étend la hiérarchie hyperarithmétique sauf qu'il doit y avoir un décalage de 1 à cause des ordinaux limites (je n'ai pas vérifié soigneusement). Cette hiérarchie M_α est la façon

dont on peut étendre le saut de Turing au-delà de ω_1^{CK} : on dit qu'une partie A de ω est un « code maître » de M_α lorsque parties de ω appartenant à $M_{\alpha+1}$ sont exactement celles qui sont Turing-réductibles à M_α : tous les M_α n'ont pas de code maître (par exemple, $M_{\omega_1^{\mathrm{CK}}}$, dont l'intersection avec $\mathscr{P}(\omega)$ est l'ensemble des parties hyperarithmétiques de ω, n'en a pas, O est le code maître de $M_{\omega_1^{\mathrm{CK}}+1}$, ce dernier ayant exactement la même intersection avec $\mathscr{P}(\omega)$), mais si on indice les M_α qui ont un code maître, on appelle ξ-ième saut de Turing absolu le (degré de Turing du) ξ-ième code maître d'un M_α. Comme ça on peut aller jusqu'à ω_1^L, qui est l'ordinal que L croit être égal à ω_1 (si $V = L$, c'est ω_1 lui-même, bien sûr). On peut relativiser tout ça (il faut définir la constructibilité relative à une partie, les M_α relatives à une partie, etc., mais il n'y a pas de difficulté fondamentale) ; il faut cependant faire attention au fait que le ξ-ième saut de Turing d'un degré de Turing d donné cesse d'être fonction croissante du degré de Turing d.

L'intersection des L_α avec l'ensemble $\mathscr{P}(\omega)$ des parties de ω prolonge quelque chose qui s'appelle la « hiérarchie analytique ramifiée » et qui est une sorte de mélange entre la hiérarchie analytique et la hiéarchie constructible. (L'idée, c'est qu'au lieu de prendre la hiérarchie analytique en considérant des quantificateurs portant sur tout ω^ω, on les prend uniquement sur les parties déjà placées dans la hiérarchie.) La hiérarchie analytique ramifiée s'arrête à un ordinal (souvent noté β_0) qui est le plus petit ordinal β tel que $L_{\beta+1}$ ne contienne aucune nouvelle partie de ω par rapport à L_β. Il se trouve que cet ordinal est le plus petit ordinal tel que L_β soit un modèle de l'axiome de séparation complet, ou que $L_\beta \cap \mathscr{P}(\omega)$ soit un modèle de l'arithmétique du second ordre. (Pour situer par rapport à ce que je dis dans la section suivante, cet ordinal β_0 est nonprojectible et beaucoup plus grand que le plus petit ordinal nonprojectible, mais beaucoup plus petit que le plus petit ordinal stable.) Je n'ai pas perdu de vue la calculabilité, ici : les parties de $\mathscr{P}(\omega)$ qui sont dans L_{β_0} sont « calculables » dans un sens très généralisé, mais pas délirant, que philosophiquement j'ai envie de décrire comme la plus grande notion possible de calculabilité qui ne fait intervenir que des parties de ω (et pas des parties de parties de ω).

14 Quelques mots sur les ordinaux récursivement grands

Ce serait dommage de ne pas profiter de l'occasion pour définir quelques ordinaux récursivement grands. (Ce sont des ordinaux dénombrables, et même inférieurs à ω_1^L, mais beaucoup plus grands que les grands ordinaux récursifs comme l'ordinal de Bachmann-Howard ou autres ordinaux intervenant dans la théorie de la démonstration comme définis par des écrasements.)

On dit qu'un ordinal λ est « récursivement inaccessible » ssi il est à la fois admissible et limite d'ordinaux admissibles ; cela revient à dire que λ est admissible et est le λ-ième

ordinal admissible.

On peut décrire le plus petit ordinal récursivement inadmissible au moyen d'un hyperoracle : c'est le plus petit ordinal qui n'est pas le type d'une relation de bon ordre sur $\mathbb{N}$ calculée par une machine de Turing ayant l'hypersaut HJ comme hyperoracle (ou, si on n'aime pas l'hypersaut, l'hyperoracle E_1 de Tugué, qui donné une suite $f = (f(0), f(1), f(2), f(3), \ldots)$ de suites finis d'entiers permet de savoir s'il existe une branche infinie, c'est-à-dire une suite infinie dont des préfixes de longueur arbitrairement grande soient dans la suite f). Et de façon analogue à ce que j'ai dit pour les machines hyperarithmétiques : une partie A de ω, ou une fonction totale $A : \omega \to \omega$, est λ-récursive avec λ le plus petit ordinal récursivement inaccessible si et seulement si A est calculable par une machine de Turing ayant HJ, ou E_1, comme hyperoracle.

On pourrait aussi parler d'ordinaux récursivement hyperinaccessibles, récursivement Mahlo, et d'autres choses comme ça, mais je laisse tomber. Ce sont des analogues assez naturelles des propriétés correspondantes pour les grands cardinaux (je crois qu'il y a aussi une notion d'ordinal « récursivement faiblement compact », mais a devient très spécialisé, là).

On dit qu'un ordinal λ est « δ-stable » (où δ est un ordinal limite quelconque, de préférence admissible) lorsque toute fonction δ-récursive, quand on lui passe des valeurs $< \lambda$, et qu'elle termine, renvoie toujours un résultat $< \lambda$ (autrement dit, λ est stable par les fonctions δ-récursives). Ceci généralise la notion d'ordinal admissible : λ est admissible ssi il est λ-stable. Un ordinal λ qui est δ-stable pour un $\delta > \lambda$ (peut-être supposé admissible) est dit « faiblement stable » : les ordinaux faiblement stables ont déjà des proprités très fortes — ils sont récursivement inaccessibles, récursivement Mahlo et plein d'autres choses. Mais il est plus intéressant de faire la définition suivante : un ordinal κ (automatiquement admissible) est dit « nonprojectible » lorsque κ est limite d'ordinaux κ-stables. Ceci n'implique pas faiblement stable (noter qu'une limite d'ordinaux δ-stables est clairement δ-stable, pour un δ fixé, mais ça ne marche plus avec « faiblement stable »), mais ça implique quand même d'être récursivement inaccessible, Mahlo, etc. Une définition équivalente est la suivante : κ admissible est nonprojectible ssi il n'existe pas de fonction κ-récursive en paramètres sur κ qui soit injective et prenne ses valeurs dans un ordinal $< \kappa$. Ou encore : ssi l'image d'une fonction κ-récursive $\kappa \to \kappa$ est bornée par un ordinal $< \kappa$, alors la fonction caractéristique de cette image est κ-récursive en paramètres. Ou encore : ssi L_κ vérifie (Kripke-Platek plus) la séparation pour les formules Σ_1 de la théorie des ensembles.

[Addendum[

J'ai été injuste envers les ordinaux nonprojectibles : je n'ai pas montré à quel point ils étaient intéressants/rigolos. Voici une façon de se rendre un peu compte des choses, et qui permet aussi d'illustrer l'importance du distinguo entre « δ-récursif sans paramètres » et « δ-récursif avec paramètres ».

Si κ est nonprojectible, alors l'ensemble des programmes pour une κ-

machine qui terminent (sans paramètres) est κ-récursif en paramètres. On est à un doigt de violer le problème de l'arrêt, là ! (En effet, l'ensemble A des programmes, vus comme des entiers naturels donc des éléments de ω, qui terminent sur une κ-machine lancée sans paramètres, est manifestement κ-récursivement énumérable, et comme il vit dans ω, il doit être κ-récursif en paramètres.)

Quelle est l'astuce ? L'astuce, c'est que si κ est nonprojectible, il y a plein d'ordinaux κ-stables $\sigma < \kappa$ (justement, κ est nonprojectible ssi il est la limite de tels σ). Si je lance une κ-machine sans paramètres, elle ne pourra jamais atteindre aucun ordinal $\geq \sigma_0$ (le plus petit ordinal κ-stable) : ça veut dire d'une part que σ_0 n'est pas κ-récursif sans paramètres, et d'autre part que si je passe σ_0 (ou n'importe quel ordinal plus grand) en argument à une κ-machine, elle peut s'en servir pour décider l'arrêt des κ-machines sans paramètres (puisqu'une κ-machine sans paramètre va faire exactement la même chose qu'une σ_0-machine). Et je pourrais continuer : si σ_1 désigne le plus petit ordinal κ-stable après σ_0, c'est aussi le plus petit ordinal qu'une κ-machine ne peut pas atteindre si on lui donne σ_0 comme paramètre (ou n'importe quels paramètres $< \sigma_1$), et recevoir σ_1 (ou tout ordinal plus grand) comme paramètre permet de décider l'arrêt des κ-machines recevant des paramètres $< \sigma_1$. Et ainsi de suite. Comme κ est limite des σ, une κ-machine a toujours le pouvoir de décider l'arrêt d'une autre κ-machine si on lui passe des arguments assez grands.

]]

Et puis, il y a les ordinaux stables : un ordinal σ est dit « stable » lorsqu'il est δ-stable pour tout δ ; i.e., toute fonction δ-récursive, pour n'importe quel δ, laisse σ stable. Il n'y a pas d'implication entre « stable » et « nonprojectible » (le plus petit ordinal stable est projectible dans ω), mais tout ordinal stable est limite de nonprojectibles, donc dans ce sens les stables sont beaucoup plus « grands ».

Le plus petit ordinal stable a un certain nombre de propriétés intressantes. Déjà, en agitant un peu les mains et en glissant un peu de poussière sous le tapis, le plus petit ordinal stable est le plus petit ordinal qui ne peut pas se définir de façon absolue. (Par exemple, s'il existe un modèle transitif de ZFC et que L_α est le plus petit tel, alors le plus petit ordinal stable σ dans l'univers est au moins égal à α, alors que dans L_α le plus petit ordinal stable est forcément $< \alpha$. Par contre, tout ordinal $< \sigma$ admet une définition absolue — genre « le plus petit α tel que L_α soit un modèle de ZFC ».)

Mais sinon, de façon plus concrète : le plus petit ordinal stable σ est le plus petit ordinal qui ne peut pas s'écrire comme le type d'un bon ordre sur les entiers qui soit Δ^1_2 dans la hiérarchie analytique ; de plus, une partie de ω, ou une fonction $\omega \to \omega$, est Δ^1_2 ssi elle est σ-calculable sans paramètres ou avec paramètres $< \sigma$, ssi elle est δ-calculable sans paramètres pour un certain ordinal δ.

(Bon, je vais arrêter là, parce que je commence à saturer.)