Calculabilité et décidabilité

1 Modèles de calcul

Au début du XX^{e} siècle le mathématicien allemand David Hilbert pose un certain nombre de problèmes à la communauté mathématique
Certains de ces problèmes n'ont pas été à ce jour résolus et ont été reconduits pour le XXI^{e} siècle
Le dixième problème de Hilbert est :
Existe t-il un **algorithme** permettant de **décider** si **toute** équation diophantienne admet une solution ?
Par exemple une équation diophantienne est $3x - 5y = 1$ avec les inconnues x et y des entiers relatifs
Une solution particulière à cette équation particulière est $x = 2$ et $y = 1$ mais il se trouve que $x = 7$ et $y = 4$ est aussi solution , on trouve donc une infinité de solutions $x = 2 + 5k$ et $y = 1 + 3k$ avec $k \in \mathbb{Z}$
Par contre l'équation $2x^2 + 2y^2 = 1$ a des solutions réelles mais pas de solutions entières
Exercice 1
A l'aide d'un argument géométrique dire pourquoi cette équation n'a pas de solutions entières mais une infinité de solutions réelles
A l'époque où le problème est posé la notion d'algorithme est "intuitive" et les mathématiciens qui se sont attaqués à ce problème ont ressenti le besoin de donner un sens mathématique à la notion d'algorithme , à la notion de calculable et par conséquent **de définir un modèle de calcul**
En 1936, dans un article intitulé *On computable numbers with an application to the entscheidungsproblem* , Alan Turing introduit un modèle (théorique) de calcul, appelée depuis lors **machine de Turing**
On peut lire cet article en ligne ici `https://www.cs.virginia.edu/~robins/Turing_Paper_1936.pdf`
(Thèse de Post) Une machine de Turing , une machine de Von Neumann, ou un langage de programmation comme Python , Java, C++, etc... sont des modèles de calcul équivalents dans le sens où "on calcule" les mêmes grandeurs avec chacun de ces modèles
D'où la définition particulière d'une fonction calculable

Définition 1 *Une fonction (au sens mathématique) est calculable si elle peut être programmée dans l'un ou l'autre des langages de programmation (Python au lycée)*

Existe-t-il des fonctions non calculables ?
Il nous reste aussi à définir le sens de "décider si toute équation diophantienne admet une solution"
A suivre ....

2 Problèmes de décision décidables

Définition 2 *Un problème de décision est la donnée d'un ensemble E d'instances et d'un sous ensemble P de E d'instances dites positives*

Exemple de problème de décision :

1. le problème de décision **Nombres premiers** a pour instances $E = \mathbb{N}$ et les instances positives est l'ensemble des nombres premiers
2. le problème de décision **Graphes connexes** a pour instances l'ensemble de tous les graphes finis et les instances positives est l'ensemble des graphes connexes
3. le problème de décision **Equations diophantiennes** a pour instances l'ensemble de **toutes** les équations diophantiennes et les instances positives est l'ensemble des équations diophantiennes ayant des soultions entières

Définition 3 *Un problème de décision (E, P) est décidable s'il existe une fonction Python qui retourne Vrai si $x \in P$ avec x quelconque dans E en entrée et faux sinon*

Exercice 2
Montrer que le problème de décision **Nombres premiers** est **décidable**
Autrement dit définir une fonction Python `estPremier(nombre)` qui retourne Vrai si `nombre` est premier Faux sinon (On ne soucie pas ici de la complexité donc on pourra définir une fonction "naïve")
Exercice 3
Montrer que le problème de décision **Graphes connexes** est **décidable** autrement dit définir une fonction Python `estConnexe(graphe)` qui retourne Vrai si `graphe` est connexe Faux sinon

1. Dans un premier temps définir une fonction naïve
2. Dans un second temps faire un parcours en largeur du graphe en partant d'un des sommets si tous les sommets ont été visités au cours de ce parcours alors le graphe est connexe

En 1970 le mathématicien russe Matiassevitch a démontré que le problème de décision **Equations diophantiennes** est **indécidable**
En quelque sorte il a montré ainsi l'existence d'une fonction non calculable, mais Turing a démontré dès 1936 l'existence d'une fonction particulière non calculable
Avant d'étudier ce problème nous allons faire un détour par les quines ...

3 Quines

Définition 4 *Dans tout langage de programmation on peut écrire un programme qui affiche son propre code. On appelle quine un tel programme (vient du nom du logicien américain William Quine)*

Exercice 4
Vérifier que le programme Python suivant est un quine

```
x = ['print("x =", x)', 'for m in x: print(m)']
print ("x =", x)
for m in x: print(m)
```

Tout ça pour insister sur le fait qu'**un programme Python est une chaine de caractères**

4 Un problème de décision célèbre : le problème de l'arrêt

Le problème de l'arrêt est défini par :
Les instances E sont toutes les fonctions Python f ayant un argument x
Les instances **positives** sont toutes les fonctions Python f ayant un argument x qui s'arrêtent au bout d'un certain moment

Théorème 1 *(Turing 1936) Le problème de l'arrêt est indécidable*

Preuve
On va faire une démonstration par l'absurde
Supposons qu'il existe un programme Python `arret(f,x)` qui retourne Vrai si $f(x)$ est un programme Python qui s'arrête avec f un programme Python quelconque et x un argument quelconque
On construit alors la fonction Python `diagonal(f)` ainsi

```
def diagonal(f):
    if arret(f,f):
        while True:
            continue
    else:
        return True
```

Ensuite on exécute `diagonal(diagonal)`
Que se passe-t-il ?
D'après notre hypothèse `arret(diagonal,diagonal)` doit nous retourner soit Vrai soit Faux
Si c'est Vrai alors après le if la fonction `diagonal(diagonal)` "entre dans une boucle infinie" **donc ne s'arrête pas** d'où une contradiction
Si c'est Faux alors la fonction `diagonal(diagonal)` **s'arrête** d'où une contradiction
On rejette donc l'hypothèse autrement dit le problème de l'arrêt est indécidable
Remarques

1. Ce type de démonstration dans l'Histoire des mathématiques porte le nom de "diagonale de Cantor" utilisée pour la première fois par le mathématicien Allemand Georg Cantor à la fin du XIX^{e} siècle, pour démontrer que $]0,1[$ n'est pas **dénombrable**
2. Tester la cohérence syntaxique de `arret(f,x)` et de `diagonal(f)` en Python (Ouvrir un environnement de travail Python)
3. La vidéo suivante est une illustration de la démonstration -> `https://www.youtube.com/watch?v=92WHN-pAFCs`