Calculabilité, décidabilité.

1 Calculabilité

Une fonction est dit **calculable** s'il existe un algorithme (une suite finie d'instructions) qui calcule cette fonction en un nombre fini d'étapes dans un ordinateur idéal, qui n'a pas de limite de temps et pas de limite d'espace.

Sinon, elle est dite incalculable.

Entre 1930 et 1936, plusieurs modèles de calculs pour formaliser les algorithmes des fonctions calculables sont inventés : les fonctions récursives de Gödel, le lamba-calcul de Church, les machines de Turing.

En 1936, la thèse de Church-Turing affirme que ces définitions sont équivalentes et une fonction calculable dans un modèle l'est dans un autre. On dit que ces modèles sont Turing-complet.

2 Fonctions incalculables

Théorème : il existe une infinité de fonction non calculable.

Démonstration

- Les programmes sont représentés en mémoire par une suite de bits. Il est donc possible d'associer à chaque programme un nombre binaire qu'il est possible de convertir en base 10.

  Ainsi pour chaque entier naturel il existe un unique programme.
- L'argument diagonale de Cantor nous dit que les nombres réels sont un ensemble plus grand que les nombres entiers, et qu'il n'est pas possible d'associer un nombre entier à chaque nombre réel.

  Les deux ensembles sont infinis mais l'ensemble des réels est un infini plus grand que l'ensemble des entiers naturels.
- Ainsi, il est impossible d'écrire chaque nombre réel un programme qui le calculerait.

Exemple de fonctions incalculables, le **castor affairé**.

3 Argument diagonal de Cantor

L'argument diagonal de Cantor est une méthode utilisée pour démontrer que certains ensembles sont non dénombrables, c'est-à-dire qu'ils ne peuvent pas être mis en correspondance biunivoque avec l'ensemble des entiers naturels. Cette méthode a été introduite par le mathématicien Georg Cantor pour prouver que l'ensemble des nombres réels est plus grand que l'ensemble des entiers naturels.

1. **Hypothèse initiale** : On suppose, pour le besoin de la démonstration par l'absurde, que l'ensemble des réels dans l'intervalle [0,1] peut être énuméré, c'est-à-dire qu'on peut associer à chaque entier naturel un réel distinct de cet intervalle.
2. **Construction d'une liste hypothétique** : On imagine qu'on a une liste infinie de réels, où chaque réel est représenté sous forme décimale. Par exemple :
   - $r_1 = 0.a_{11}\ a_{12}\ a_{13}\ \ldots$
   - $r_2 = 0.a_{21}\ a_{22}\ a_{23}\ \ldots$
   - $r_3 = 0.a_{31}\ a_{32}\ a_{33}\ \ldots$
   - Et ainsi de suite.
3. **Construction du réel diagonal** : On crée maintenant un nouveau nombre réel en choisissant, pour chaque décimale de la n-ème position, un chiffre qui est différent de celui de la n-ème ligne. Autrement dit, si le n-ème chiffre de r_n est a_{nn}, on choisit un chiffre b_n qui est différent de a_{nn}.
4. **Contradiction** : Le nombre ainsi construit ne peut être dans la liste initiale, car il diffère de chaque r_n par au moins un chiffre (le n-ème chiffre). Cela signifie qu'il existe un réel qui ne figure pas dans la liste, ce qui contredit l'hypothèse initiale selon laquelle tous les réels de [0,1] étaient énumérés.
5. **Conclusion** : L'ensemble des réels n'est pas dénombrable, et donc il existe des ensembles dont la cardinalité est strictement supérieure à celle des entiers naturels.

$$
\begin{array}{|ll|}
\hline
s_1 & = 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ \ldots \\
s_2 & = 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ \ldots \\
s_3 & = 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ \ldots \\
s_4 & = 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ \ldots \\
s_5 & = 1\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ \ldots \\
s_6 & = 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ \ldots \\
s_7 & = 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ \ldots \\
s_8 & = 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ \ldots \\
s_9 & = 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ \ldots \\
s_{10} & = 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ \ldots \\
s_{11} & = 1\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ \ldots \\
\vdots & \quad\ \vdots\ \vdots\ \vdots\ \vdots\ \vdots\ \vdots\ \vdots\ \vdots\ \vdots\ \vdots\ \vdots\ \ddots \\
\hline
\end{array}
$$

$$
\boxed{s \ = 1\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ \ldots}
$$

4 Problème de l'arrêt

Les fonctions dont la réponse est soit oui soit non sont appelés problèmes de décision. Pour les problèmes de décision on utilise les termes décidables et indécidables à la place de calculable et incalculable.

Le problème de l'arrêt est le problème de décision qui détermine, à partir d'une description d'un programme informatique, et d'une entrée, si le programme s'arrête avec cette entrée ou non.

Alan Turing a montré en 1936 que le problème de l'arrêt est indécidable, c'est-à-dire qu'il n'existe pas de programme informatique qui prend comme entrée une description d'un programme informatique et un paramètre et qui, grâce à la seule analyse de ce code, répond VRAI si le programme s'arrête sur son paramètre et FAUX sinon. Une partie importante de la démonstration a été la formalisation du concept de programmes informatiques : les machines de Turing.

Démonstration

Donnons ici la preuve de ce résultat fondée sur l'idée utilisée par Turing dans son article fondateur de 1936 (page 247). Elle repose sur un argument diagonal, tout comme la preuve d'indénombrabilité des réels de Cantor (1891) et celle du théorème d'incomplétude de Gödel (1931).

Sans entrer dans le formalisme des machines de Turing, il suffit de savoir que toute machine de Turing (autrement appelée programme) prend en entrée des mots finis `M` sur un alphabet fini. Par ailleurs, on décrit un programme (machine de Turing) par un mot fini `PROG`, qui représente son codage.

Montrons par l'absurde que le problème de l'arrêt est indécidable.

1. **Hypothèse initiale** : Supposons qu'il existe un programme `HALT` qui décide le problème de l'arrêt. C'est-à-dire :

   Si `PROG(M)` s'arrête, alors `HALT(PROG, M)` renvoie `VRAI` en un temps fini ;

   Si `PROG(M)` ne s'arrête pas, alors `HALT(PROG, M)` renvoie `FAUX` en un temps fini.

2. **Construction du réel diagonal** : On construit le programme diagonale suivant :

```
FONCTION diagonale(x):
     si HALT(x, x)
          boucle infinie
     sinon
          renvoie VRAI
```

3. **Contradiction :** On obtient une contradiction pour l'entrée `diagonale`.

   `diagonale(diagonale)` boucle à l'infini si et seulement si `HALT(diagonale, diagonale)` renvoie `VRAI`, c'est-à-dire si et seulement si `diagonale(diagonale)` termine.

4. **Conclusion** : Cela prouve donc par l'absurde que `HALT` n'existe pas.