Calculabilité, Complexité,

1 – Quelques notions intuitives

P. Berthomé

INSA Centre Val de Loire
Département STI

22 septembre 2020

De quoi va-t-on parler ici ?

Objets du cours

- Problèmes et algorithmes
- Complexité
- Calculabilité

Dans quel but ?

- Comprendre la hiérarchie des problèmes, si possible identifier les problèmes difficiles
- Comprendre la puissance de l'informatique et ses limites

Intérêt pour un ingénieur STI ?

- Avoir une culture théorique minimale
- Être capable de trouver la meilleure solution : la meilleure façon n'est pas nécessairement la plus simple.

Buts de ce module

Formalisation d'algorithmes et de problèmes : calculabilité

- Savoir ce qui est fondamental
- Ce que l'on peut faire
- Ce que l'on ne peut pas faire
- Définition d'un modèle de calcul minimal, mais universel : les machines de Turing
- Limites avec ce modèle : théorème de la Halte
- Autres modèles

Notions de complexité

- Machines de Turing Non-Déterministes
- Problèmes NP-Complets
- . . . Approximations de problèmes

Problèmes, Instances

Definition

Un **problème** est une question générique sur un **domaine**. Une **instance** d'un problème est la question posée pour un élément du domaine. Quand la réponse à un problème est binaire (Oui/Non), c'est un **problème de décision**.

Example

- Déterminer si un entier est pair
  - Problème dont le domaine est $\mathbb{N}$
- Déterminer si 7853 est pair
  - Instance du problème précédent
- Trier un tableau d'entiers
- Déterminer si un programme écrit en C s'arrête sur n'importe quelle entrée

Algorithme

Definition

Un **algorithme** est un procédé qui aboutit à un résultat. Les algorithmes manipulent des mots sur un alphabet fini Σ

Example

- Calcul d'une fonction sur les entiers
- Construction d'un objet

Première notion de calculabilité

Hypothèse

Tout algorithme est décrit comme un texte (alphabet fini Σ)

Combien d'algorithmes ?

Infini,oui mais dénombrable (autant que d'entiers)

Combien de fonctions : $\mathbb{N} \longrightarrow \mathbb{N}$?

Infini, oui mais non dénombrable (en gros, autant que de réels)

Conclusion

- Il existe des fonctions $\mathbb{N} \longrightarrow \mathbb{N}$ qui ne sont pas calculables par un algorithme (Voir TD)
- Définir n'est pas synonyme de calculer

Thèse de Church

Thèse de Church

On ne peut cerner la notion de calculabilité par algorithme de manière algorithmique.

Historique

Diverses notion équivalentes données au cours du XXe siècle

- Ensembles récursifs (Church, Gödel, Von Neumann)
- Machines de Turing (Turing-1936)
- Programmes RAM

Machines de Turing

Concepts

- Automates –> Langages réguliers
- Automates à Piles –> Langages algébriques
- Machines de Turing –> langages *calculables*

Intérêt

- Trouver le modèle le plus simple pour décrire tout algorithme
- Pas forcément le plus efficace
- Machine idéale avec des ressources infinies.

Composants de la Machine de Turing

Alphabet Σ

On travaille avec un alphabet fini Σ qui contient un symbole particulier **B** (blanc).

Rubans

Succession bi-infinie de cases sur lesquelles sont écrites des lettres de Σ

Tête de lecture-écriture (*RWH*)

Élément positionné sur une case du ruban qui :

- lit la lettre sur la case où elle se trouve
- écrit une lettre sur cette même case
- se déplace à droite, à gauche ou reste au même endroit

Composants de la Machine de Turing

Ensemble d'états

Ensemble **fini** Q avec :

- q_0 état initial
- $Q_f \subseteq Q$ les états finaux séparé en deux : Q_A et Q_R

Fonction de transition

Fonction qui décide à partir de :

- l'état q dans laquelle se trouve la machine
- les lettres lues par les RWH

de

- son nouvel état
- des lettres à écrire par chacune des RWH
- mouvement de chacune des RWH (D, G, St)

Fonctionnement de la Machine de Turing

Temps discret

Une MT fonctionne suivant un temps discret (étape par étape)

Configuration initiale

- Toutes les cases sont blanches sauf un nombre fini d'entre elles : c'est le **mot d'entrée**
- Toutes les RWH d'un ruban sont sur la case blanche immédiatement à gauche de la 1ère case non blanche
- L'état est q_0

Fonctionnement de la Machine de Turing

À chaque étape

- Elle applique la fonction de transition

Terminaison (ou pas)

1. Si elle entre dans un état final (Q_f), la machine s'arrête
   - si $q \in Q_A$, elle accepte le mot
   - sinon elle refuse le mot
2. Si elle n'entre jamais dans un état de Q_f, la machine ne s'arrête pas

Langage reconnu

Ensemble des mots d'entrée pour lesquels la machine de Turing s'arrête sur un état d'acceptation.

Une machine de Turing, en bref

7-uplet : $(\Sigma, B, Q, q_0, Q_f, m, \delta)$

- Σ alphabet fini et B une lettre de Σ
- Q un ensemble fini d'états, q_0 l'état initial
- Q_f l'ensemble des états finaux
- m une suite finie d'entiers strictement positifs :
  - $m = (n_1, n_2, \ldots, n_{|m|})$
  - $|m|$ est le nombre de rubans
  - n_i est le nombre de RWH sur le i-ème ruban
  - $||m|| = \sum_{i=1}^{|m|} n_i$
- $\delta : Q \times \Sigma^{||m||} \longrightarrow Q \times \Sigma^{||m||} \times \{G, St, D\}^{||m||}$

Description instantanée

Etat d'une MT plus le ruban à un instant donné.

Que faire avec une Machine de Turing

Des calculs

- Des opérations sur des mots
- Par exemple, faire la somme de deux entiers
- Idées ?

Reconnaissance d'ensembles/de langages

- Pour $x \in (\Sigma \setminus \{B\})^*$ et une relation $\mathcal{R} \subseteq (\Sigma \setminus \{B\})^*$
- Trouver une machine de Turing MT qui prend en entrée x et s'arrête sur un état d'acceptation si $x \in \mathcal{R}$
- **Ensemble décidable** : il existe une MT, telle que pour tout x, MT dit que $x \in \mathcal{R}$ ou $x \notin \mathcal{R}$
- **Ensemble semi-décidable** : il existe une MT, telle que pour tout élément x de $\mathcal{R}$, MT dit que $x \in \mathcal{R}$.