Introduction à la calculabilité

Limites de ce qu'un algorithme peut faire.

Σ alphabet fini (p. ex. $\Sigma = \{0, 1\}$ ou $\Sigma =$ caractères ASCII Unicode)

Not: suite finie de symboles de Σ (Σ^* ensemble de ts les mots)

Langage: ensemble de mots (élément de $\mathcal{P}(\Sigma^*)$).

Un problème de décision peut être vu comme un langage de tous les mots correspondant à des instances vraies du problème).

Par exemple, le problème SAT peut être vu comme le langage de toutes les formules de la logique propositionnelle qui sont satisfiables.

Réciproquement, tout langage est un problème de décision : le problème de décider si un mot appartient au langage.

Algorithme : procédure de calcul implémentable

Déf. (semi-décidabilité, récursive énumérabilité)

On dit qu'un langage L est semi-décidable ou récursivement énumérable s'il existe un algorithme A qui énumère, l'un après l'autre, tous les mots du langage.

Ex. $L = \{0^n \mid n \geq 0\}$

A :
```
n := 0
Tant que Vrai
    Produire 0^n
    n := n + 1
```

Prop. Un langage L est semi-décidable ssi il existe un algorithme A qui prend en entrée un mot u de Σ^* et :

(*) — si $u \in L$ alors A renvoie Vrai
— si $u \notin L$ alors soit A renvoie Faux, soit A ne répond pas (boucle infinie)

Dém. Soit L semi-décidable. Il existe un algorithme A qui l'énumère. On construit un algorithme A' prenant en entrée un mot u et lançant l'algorithme A jusqu'à ce que A produise u, puis renvoyant Vrai.

Soit A un algorithme avec les conditions (*).
On construit un algorithme A' d'énumération des mots de L. (dovetailing) :

* On énumère un mot $u \in \Sigma^*$ en utilisant un algorithme qui énumère tous les mots
* On lance la première étape de calcul de A sur u. Pour tous les mots v énumérés jusqu'à présent, on lance une étape de calcul supplémentaire.
* Si l'un des calculs sur un mot v se termine en produisant Vrai, alors on énumère v.
* On revient à la première étape. □

Déf. Un langage L est <u>récursif</u> ou <u>calculable</u> ou <u>décidable</u> s'il existe un algorithme A prenant en entrée un mot $u \in \Sigma^*$ et :
- renvoyant Vrai si $u \in L$
- renvoyant Faux sinon.

Prop. Un langage L est décidable ssi L est semi-décidable et $\overline{L}$ est semi-décidable.

$\overline{L}$: langage complémentaire

Il existe des langages (beaucoup !) non semi-décidables.
(Compter les langages, compter les algorithmes).

Théorème (Cantor) Soit E un ensemble arbitraire.
Il n'existe pas de surjection de E vers $\mathcal{P}(E)$.

Démonstration : Supposons par l'absurde qu'il existe $f : E \to \mathcal{P}(E)$ surjective ! On pose $X = \{e \in E \mid e \notin f(e)\}$.
$X \in \mathcal{P}(E)$. Donc il existe $x \in E$ t.q. $f(x) = X$.

- Supposons $x \in X$. Donc $x \notin f(x) = X$. Contradiction.
- Supposons $x \notin X = f(x)$. Alors $x \in X$. Contradiction. □

Supposons que tous les problèmes soient semi-décidables.
Ça veut dire qu'il existe une fonction surjective :

Ens. des algorithmes $\longrightarrow$ Ens. des problèmes

↑ décrits par des suites finies de bit (∞-ens. de)

Σ^* dénombrable $\longrightarrow$ $\mathcal{P}(\Sigma^*)$ non dénombrable

Un algorithme peut être vu comme un mot de Σ^+.
Un problème — un langage sur Σ, c'est-à-dire un ensemble de mots de Σ^+.

Impossible par le th. de Cantor !

Il existe des pb semi-décidables mais non décidables.

Entscheidungsproblem (pb de décision) : trouver une mécanisation du calcul permettant, étant donné un énoncé mathématique, de déterminer s'il est vrai ou faux.

Théorème (Alan Turing) Le problème de l'arrêt est semi-décidable mais non décidable.

$$H = \{ (\text{algorithme, données d'entrées}) \mid \text{l'algorithme termine sur ces données} \}$$

Dém. H est semi-décidable.

On construit un algorithme B qui prend en entrée un mot $x \in \Sigma^*$ et :

- si x n'est pas de la forme (algorithme, données), renvoie faux.
- sinon, $x = (A, u)$. On applique A à u. Si A termine alors on renvoie Vrai.

On montre que H n'est pas décidable. Supposons par l'absurde que H est décidé par B.

Les algorithmes et les données d'entrées sont dénombrables donc on peut les numéroter

$(A_0, A_1, A_2 \dots)$

$(u_0, u_1, u_2 \dots)$,

On construit un algorithme B' prenant en entrée un mot $u_i \in \Sigma^*$:

- On applique B à (A_i, u_i)
- Si B renvoie Faux, on renvoie Faux.
- Sinon, on part dans une boucle infinie.

B' étant un algorithme, il existe un k t.q. $B' = A_k$.

Appliquons B à $(B' = A_k, u_k)$

* Si B renvoie Vrai, ça veut dire que $A_k(u_k)$ termine. Mais $A_k(u_k)$ c'est $B'(u_k)$, et par définition de B', $B(A_k, u_k)$ renvoie Faux.

* Si B renvoie Faux, ça veut dire que $A_k(u_k)$ part dans une boucle infinie. Ça veut dire que $B(A_k, u_k)$ renvoie Vrai. □

Ex. de pb semi-décidables mais pas décidable :

- pb de l'arrêt
- déterminer si une fonction implémentée dans un langage de programmation classique a n'importe quelle propriété non triviale à l'exécution (théorème de Rice)
- étant donnés deux ensembles finis de mots, déterminer si une séquence de mots du premier ensemble est identique à une séquence du deuxième ensemble (correspondance de Post)
- étant donnée des règles de calcul dans un groupe, déterminer si deux expressions sont égale
- déterminer si une formule de logique des prédicats est satisfiable

...

Ex. de langage non semi-décidable

- langage des (programmes, données) t.q. le programme ne s'arrête pas sur ces données.

Modèles de calcul

Def. Une machine de Turing (déterministe) est défini par :

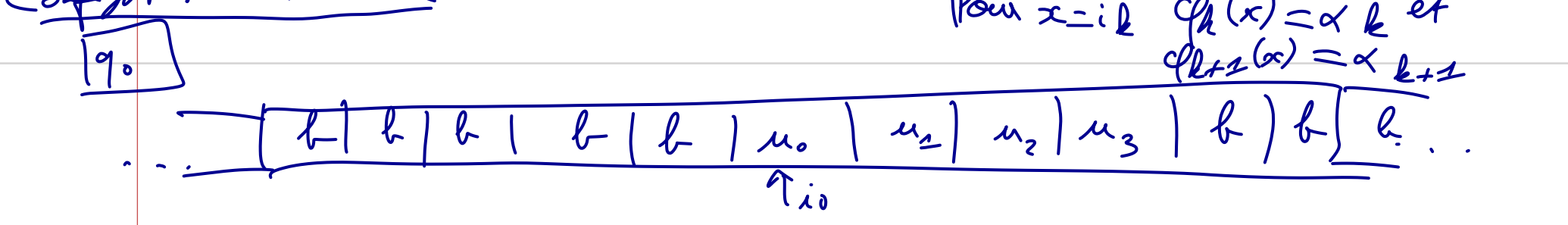
- 1/ un ensemble fini d'états Q
- 2/ un alphabet de travail Γ (ensemble fini)
- 3/ $b \in \Gamma$ est un symbole spécial ("blanc")
- 4/ $\Sigma \subseteq \Gamma \setminus \{b\}$ est l'alphabet d'entrée - sortie
- 5/ $q_0 \in Q$ est un état initial
- 6/ $F \subseteq Q$ est un ensemble d'états finaux
- 7/ $\delta : Q \setminus F \times \Gamma \to Q \times \Gamma \times \{L, R\}$

(automates) (x)

Etant donnée une machine de Turing, un calcul sur l'entrée $u \in \Sigma^+$ est une séquence finie de configurations $\{(q_k, \varphi_k, i_k)\}_{1 \le k \le n}$ où $q_k \in Q$, $\varphi_k : \mathbb{Z} \to \Gamma$ est une fonction associant à une bande virtuelle infinie, et $i_k \in \mathbb{Z}$ est la position actuelle sur la bande. On impose les conditions suivantes :

- $q_1 = q_0$, $i_1 = 0$, $\varphi_1(x) = u_x$ pour $1 \le x \le |u|$ et $\varphi_1(x) = b$ pour $x \notin [1, |u|]$
- Pour $1 \le k \le n-1$, $\delta(q_k, \alpha_k) = (q_{k+1}, \alpha_{k+1}, \beta)$ avec
  - $i_{k+1} = i_1 + 1$ si $\beta = R$, -1 si $\beta = L$
  - Pour $x \neq i_k$ $\varphi_k(x) = \varphi_{k+1}(x)$
  - Pour $x = i_k$ $\varphi_k(x) = \alpha_k$ et $\varphi_{k+1}(x) = \alpha_{k+1}$

Configuration initiale



Un calcul est acceptant (renvoie Vrai) s'il conduit à un état final, échouant (renvoi Faux) sinon.

Théorème de Church-Turing

En terme de modèle de calcul, les formalismes suivants sont équivalents :

- Machine de Turing (déterministe ou non-déterministe)
- Lambda-calcul
- Fonctions récursives de Kleene
- Langages de programmation usuels (sans contrainte d'environnement) (Turing-complet)
- Machine à registre

Thèse de Church-Turing

Tout ce qui est (intuitivement / humainement / naturellement) calculable, est calculable au sens des machines de Turing.