1 Objectifs

1. Que peut-on «reconnaître» ? ... au-delà des langages...
   - ... réguliers (facile)
   - ... hors-contexte (pas trop difficile)
   - ... sous-contexte (encore faisable...)
2. Que peut-on «calculer» ?
   (quelles fonctions de $A \to B$ peut-on *programmer* ?)
3. Que peut-on «décider» ? ... à propos d'un problème
   (fonction totale $\{$instances du problème$\} \to \{$oui/non$\}$)

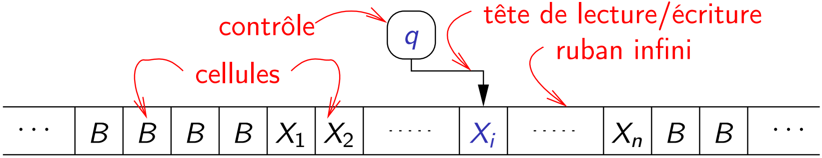
On verra que 1. $\Leftrightarrow$ 2. $\Leftrightarrow$ 3.

- On verra qu'on ne peut pas tout reconnaître/calculer/décider
- On verra comment on peut reconnaître/calculer/décider
  et comment prouver qu'on ne peut pas

On abordera aussi 2. sous l'angle de la complexité
Plutôt que d'étudier tout ça en Python, C, Java... on adopte un formalisme simplissime (mais autant expressif, et historique...) :
les *Machines de Turing* [1936, Alan M. Turing (1912-1954)]

2 Machines de Turing

Une machine de Turing (MT) nécessite pour être «exécutée» :



- *mémoire* : *ruban* (bande) infini découpé en *cellules* (cases)
  chaque cellule contient un symbole d'un *alphabet du ruban* Γ
  un nombre *fini* (mais non borné) de cellules est «occupé »
  les autres sont «vides» (symbole : B — «blanc»)
  $\leadsto$ s'affranchir de «mémoire bornée»
- *tête de lecture/écriture* : «pointe» sur une case du ruban
- *partie contrôle* : contient un *état* q (idem AFD)

Configuration (de la MT) :

$<$contenu du ruban, position de la tête, état du contrôle$>$

L'*évolution* (l'éxécution) de la MT dépend uniquement de q et X_i, et est définie par (le programme de) la MT

Une MT est un septuplet $m = (Q, \Gamma, \Sigma, B, q_0, F, \delta)$ où :

- Q est un ensemble (fini) d'*états*
- Γ est un vocabulaire (*alphabet du ruban*)
- $\Sigma \subset \Gamma$ est l'*alphabet d'entrée* $(\Sigma \neq \emptyset)$
- $B \in \Gamma - \Sigma$ est le « *blanc* »
- $q_0 \in Q$ est l'*état initial*
- $F \subseteq Q$ est l'ensemble des *états finals* (états accepteurs)
- $\delta : (Q \times \Gamma) \longrightarrow (Q \times \Gamma \times \mathcal{M})$ (avec $\mathcal{M} = \{G, D, S\}$) est la *fonction (partielle) de transition*

$\delta(q, X) = (p, Y, M)$: que faire si le contrôle est dans l'état q et le symbole sous la tête de lecture/écriture est X

- p : nouvel état ($p = q$ possible)
- Y : remplacement de X par Y ($Y = X$ possible)
- M : mouvement de la tête de lecture
  (vers la *G*auche, vers la *D*roite, *S*tationnaire)

Configuration : ruban : $B^\infty \alpha X \beta B^\infty$, tête sur X, état q
notée $\boxed{\alpha q X \beta}$ $(\alpha, \beta \in \Gamma^*, X \in \Gamma)$

Exécution : *transitions* (évolution, changement de configuration)

$$\boxed{\alpha q X \beta \vdash \alpha' q' X' \beta'}$$

déterminée par $\delta(q, X)$

- $\delta(q, X) = (p, Y, S)$: $\alpha q X \beta \vdash \alpha p Y \beta$
- $\delta(q, X) = (p, Y, D)$:
  - si $\beta = Z\beta'$: $\alpha q X \beta = \alpha q X Z \beta' \vdash \alpha Y p Z \beta'$
  - si $\beta = \varepsilon$: $\alpha q X \equiv \alpha q X B \vdash \alpha Y p B$
- $\delta(q, X) = (p, Y, G)$:
  - si $\alpha = \alpha' Z$: $\alpha q X \beta = \alpha' Z q X \beta \vdash \alpha' p Z Y \beta$
  - si $\alpha = \varepsilon$: $q X \beta \equiv B q X \beta \vdash p B Y \beta$
- $\delta(q, X)$ non défini : *configuration terminale* : $\alpha q X \beta \not\vdash$

Un mot $w \in \Sigma^*$ est $\boxed{\textit{accepté}}$ par une MT $m \Leftrightarrow$

$$q_0 w \vdash^* \alpha p X \beta \not\vdash \text{ avec } p \in F \text{ (état final)}$$

$\rightarrow$ dans l'état initial, avec w sur le ruban et la tête au début de w,
on atteint une configuration terminale dans un état final

Langage $\boxed{\text{accepté}}$ par une MT m :

$$L(m) = \{w \in \Sigma^* \ : \ w \text{ accepté par } m\}$$

Un langage L est dit $\boxed{\text{récursivement énumérable}}$ (r.e.) s'il existe une MT m qui l'accepte : $L = L(m)$

L'ensemble des langages r.e. est noté RE

On peut montrer que RE est exactement l'ensemble des langages de type 0 (engendrés par des grammaires quelconques)

Une MT peut ne pas s'arrêter (ne pas atteindre une configuration terminale) sur certaines entrées w

Une MT qui s'arrête $\forall w \in \Sigma^*$ permet de $\boxed{\text{décider}}$ si $w \in L(m)$:

- si $q_0 w \vdash^* \alpha p X \beta \not\vdash$ avec $p \in F$ (état final) : OK
- si $q_0 w \vdash^* \alpha p X \beta \not\vdash$ avec $p \notin F$ (état non final) : KO

Un langage L est dit $\boxed{\text{récursif}}$ s'il est *décidé* par une MT

L'ensemble des langages récursifs est noté R

On a évidemment $R \subseteq RE$

Note : on peut toujours transformer une MT pour avoir
$F = \{f\}$ et $\delta(f, X)$ non défini $\forall X$ (avec f nouvel état) :
$\forall q \in F_{orig} : \forall X \in \Gamma$:

$\delta(q, X)$ non défini $\rightsquigarrow$ ajouter $\delta(q, X) = (f, X, S)$

Exemples de machines de Turing

$L = \{a^n b^p\} = a^* b^* \quad (\Sigma = \{a, b\})$
$m = (\{q_0, q_1, f\}, \{a, b, B\}, \{a, b\}, B, q_0, \{f\}, \delta)$

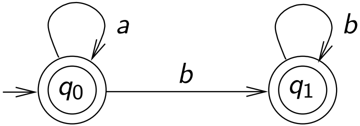
1. $\delta(q_0, a) = (q_0, a, D)$ // avancer tant qu'on a des a
2. $\delta(q_0, B) = (f, B, S)$ // on est au bout : OK
3. $\delta(q_0, b) = (q_1, b, D)$ // on rencontre un b, a interdit
4. $\delta(q_1, b) = (q_1, b, D)$ // avancer tant qu'on a des b
5. $\delta(q_1, B) = (f, B, S)$ // on est au bout : OK

$$q_0 aabbb \vdash^* aabbq_1 b \vdash aabbbq_1 B \vdash aabbbfB \not\vdash \text{OK}$$
$$q_0 aabba \vdash^* aabbq_1 a \not\vdash \text{KO}$$

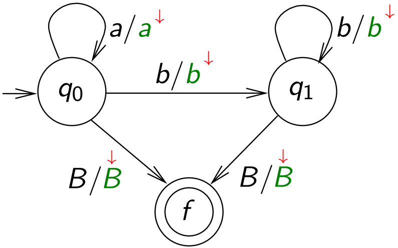
Langages réguliers : facile, toujours possible d'avancer au plus «au-delà du mot» (sur le B qui suit) pour décider

Une notation graphique pour les MT

L'AFD (incomplet)



La MT



$L = \{a^n b^n\}$

Facile en récursif (LL(1)) ou en itératif (compter jusqu'à n)

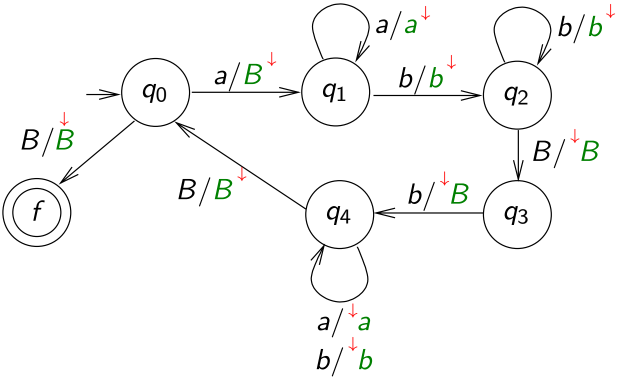
Notre modèle nous fournit les deux : pile infinie, entiers non bornés (codés avec un nombre fini de symboles, p.ex. 0 et 1 !)

Mais compliqué...

Autre idée : on «efface» le a initial
et le b final «correspondant»...

1. $\delta(q_0, B) = (f, B, S)$ // mot vide : fini OK
2. $\delta(q_0, a) = (q_1, B, D)$ // effacer le a et avancer
3. $\delta(q_1, a) = (q_1, a, D)$ // avancer...
4. $\delta(q_1, b) = (q_2, b, D)$ // ... jusqu'à avoir trouvé un b
5. $\delta(q_2, b) = (q_2, b, D)$ // continuer jusqu'à...
6. $\delta(q_2, B) = (q_3, B, G)$ // ... trouver un blanc puis reculer
7. $\delta(q_3, b) = (q_4, B, G)$ // effacer le b et reculer
8. $\delta(q_4, a) = (q_4, a, G)$ // ... jusqu'à...
9. $\delta(q_4, b) = (q_4, b, G)$ // ... trouver un...
10. $\delta(q_4, B) = (q_0, B, D)$ // ... blanc... et on recommence

Version graphique...



3 Tout est naturel

Σ^* peut toujours être vu comme un *codage* bijectif de $\mathbb{N}$:

$w = \tilde{n}$	ε	a	b	c	aa	ab	ac	ba	bb	bc	...
$n = <w>$	0	1	2	3	4	5	6	7	8	9	...

Codage *effectif* : $\exists$ des algos $w \mapsto <w>$ et $n \mapsto \tilde{n}$

Exemple : $\text{ZOU}^* = \{0,1\}^*$:

$w = \tilde{n} = w_n$	ε	0	1	00	01	10	11	000	001	...
$n = <w>$	0	1	2	3	4	5	6	7	8	...
n(binaire)	0	1	10	11	100	101	110	111	1000	...

$<w> = (1w - 1)$: insérer «1» en tête, puis soustraire 1 en binaire

$\tilde{n} = (n+1)$ dont on supprime le «1» de tête $\rightsquigarrow w_n$

Donc «langage sur Σ» = «sous-ens. de Σ^*» $\equiv$ «sous-ens. de $\mathbb{N}$»

On parle d'*ensembles* récursifs, récursivement énumérables...

... quand on s'intéresse aux sous-ensembles de $\mathbb{N}$...

... ou aux sous-ensembles de tout ensemble codage bijectif de $\mathbb{N}$

Exemples : $\mathbb{Z}$, $\mathbb{N}^\star$, $\mathbb{N}^2$, ${\Sigma_1}^* \times {\Sigma_2}^*$, $\mathbb{N}^k$, $\mathbb{N}^*$, $(\mathbb{N}^2)^*$...

Note : les ensembles finis ($\equiv \mathcal{P}_f(\mathbb{N})$) sont évidemment récursifs

4 Fonctions calculables

Pour une MT m donnée, quand un mot w $(= \tilde{n})$ est accepté,
le ruban «contient» un mot w' de Γ^*,
qui peut être aussi considéré comme un certain $\tilde{p}$...

Le blanc B pose problème pour *définir* w' en général
mais on peut restreindre à un w''...
p.ex. : symboles entre tête de lecture et le premier B (exclu)...

La MT m est alors vue comme calculant une fonction

$$f : \begin{array}{ccc} \mathbb{N} & \rightarrow & \mathbb{N} \\ n & \mapsto & p \end{array}$$

f peut ne pas être totale : $\mathrm{dom}(f) = \{n : \tilde{n} \text{ accepté}\}$

Définition : une fonction f est (partielle) $\boxed{\text{calculable}}$ ssi
$\exists m \in \mathrm{MT} : m$ calcule f

Définition : une fonction f est $\boxed{\text{totale calculable}}$ ssi
$\exists m \in \mathrm{MT} : m$ calcule f et $\mathrm{dom}(f) = \mathbb{N}$

f peut évidemment être une fonction de $A \rightarrow B$
où A et B : codages bijectifs de $\mathbb{N}$

Les fonctions partielles permettent également d'avoir A fini...

Pour A et B on peut choisir le codage qu'on veut

$\mathbb{N}$: en unaire (4 : 1111), binaire (4 : 100) ou autre

$\mathbb{N}^2$: (n, p) codé par $\tilde{n}\#\tilde{p}$ ... ou autre

Lien entre RE, R et fonctions calculables

$E \in \mathrm{RE} \Leftrightarrow E$ est le domaine d'une fonction calculable

$E \in \mathrm{R} \Leftrightarrow$ la *fonction caractéristique* de E est totale calculable

Fonction caractéristique de E : $1_E(n) = 1$ si $n \in E, 0$ sinon

Il existe *évidemment* des fonctions non calculables...

$\{f : \mathbb{N} \to \mathbb{N}\}$ a la puissance du continu (n'est pas dénombrable)

Une MT m *est un programme*, écrit dans un *langage* des MT

... qu'on peut définir formellement (comme tout langage)

... sur un certain alphabet Σ_{MT}

Une MT m est donc un élément de $(\Sigma_{MT})^*$ qui est dénombrable...

Il existe donc une infinité (continue) de fonctions non calculables ■

Thèse de Church-Turing (une de ses formulations) :

«toute fonction "*intuitivement calculable*" (pour laquelle on peut imaginer un "algorithme") est "*effectivement* calculable" (programmable, entre autres par une MT)»