IF228 - Calculabilité et Complexité

Cours #2 : machines universelles, thèse de Church-Turing

Frédéric Herbreteau
frederic.herbreteau@bordeaux-inp.fr
(d'après les supports de Corentin Travers)

8 janvier 2025

Robustesse du modèle de calcul par Machines de Turing, Machine de Turing universelle

Plan

1. Robustesse du modèle calcul par Machine de Turing
2. Codage des machines de Turing
3. Machine de Turing universelle
4. Thèse de Church-Turing

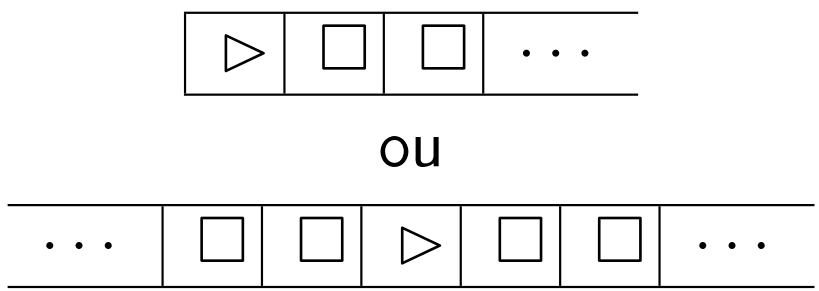
$\longrightarrow$ se convaincre que les MTs sont un **bon modèle d'ordinateur**

Robustesse du modèle de calcul par Machines de Turing

Ou : “pourquoi les détails de la définition n’importent pas”

Des choix arbitraires

- Nombre de rubans : 1,2,. . .,k
- Choix de l'alphabet : $\{0, 1\}$, $\{a, b, c, \ldots, z\}$, etc.
- Rubans infinis dans une direction ou deux directions :



Taille de l'alphabet

Soit $f : \{0,1\}^* \to \{0,1\}^*$ et $T : \mathbb{N} \to \mathbb{N}$

Lemme

*Si f est calculable par une **MT** $M = (\Sigma, Q, q_{start}, F, \delta)$ en temps T, alors f est calculable par une MT M' définie sur **l'alphabet** $\{0,1,\square,\triangleright\}$ en temps* $4\log(|\Sigma|)T$.

Taille de l'alphabet

Preuve

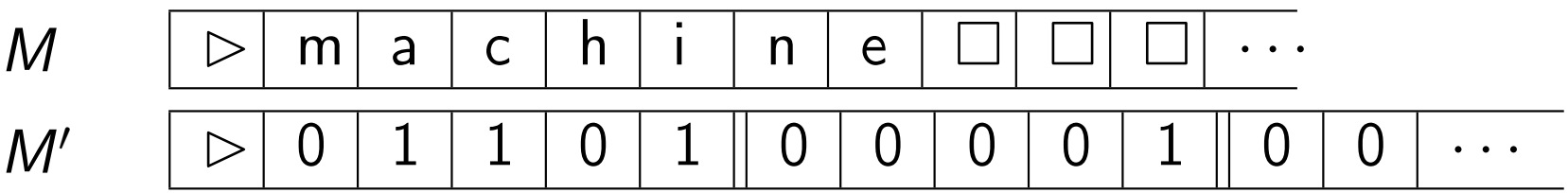
But : simuler $M = (\Sigma, Q, q_{start}, F, \delta)$ par $M' = (\{0, 1, \square, \triangleright\}, Q', q'_{start}, F', \delta')$

Taille de l'alphabet

Preuve

But : simuler $M = (\Sigma, Q, q_{start}, F, \delta)$ par
$M' = (\{0, 1, \square, \triangleright\}, Q', q'_{start}, F', \delta')$

- Coder chaque symbole de Σ sur $k = \lceil \log |\Sigma| \rceil$ bits

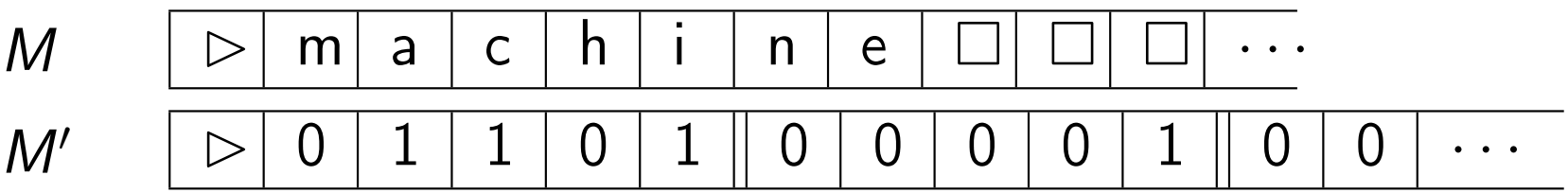


Taille de l'alphabet

Preuve

But : simuler $M = (\Sigma, Q, q_{start}, F, \delta)$ par $M' = (\{0, 1, \square, \triangleright\}, Q', q'_{start}, F', \delta')$

- Coder chaque symbole de Σ sur $k = \lceil \log |\Sigma| \rceil$ bits



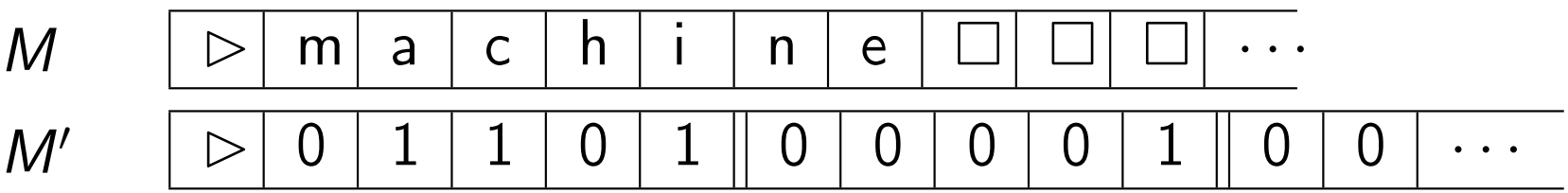
- simuler lecture/écriture des symboles faites par M : k transitions de M', utiliser l'état de M' pour stocker les symboles lus/à écrire

Taille de l'alphabet

Preuve

But : simuler $M = (\Sigma, Q, q_{start}, F, \delta)$ par $M' = (\{0, 1, \square, \triangleright\}, Q', q'_{start}, F', \delta')$

- Coder chaque symbole de Σ sur $k = \lceil \log |\Sigma| \rceil$ bits



- simuler lecture/écriture des symboles faites par M : k transitions de M', utiliser l'état de M' pour stocker les symboles lus/à écrire

- $|Q'| = c|Q||\Sigma'|^k$ où c est une constante

Nombre de rubans

Une MT à un seul ruban ne possède qu'un seul ruban qui sert à la fois de ruban d'entrée, de sortie et de travail et qui est accessible en lecture et écriture

Soit $f : \{0,1\}^* \to \{0,1\}^*$ et $T : \mathbb{N} \to \mathbb{N}$

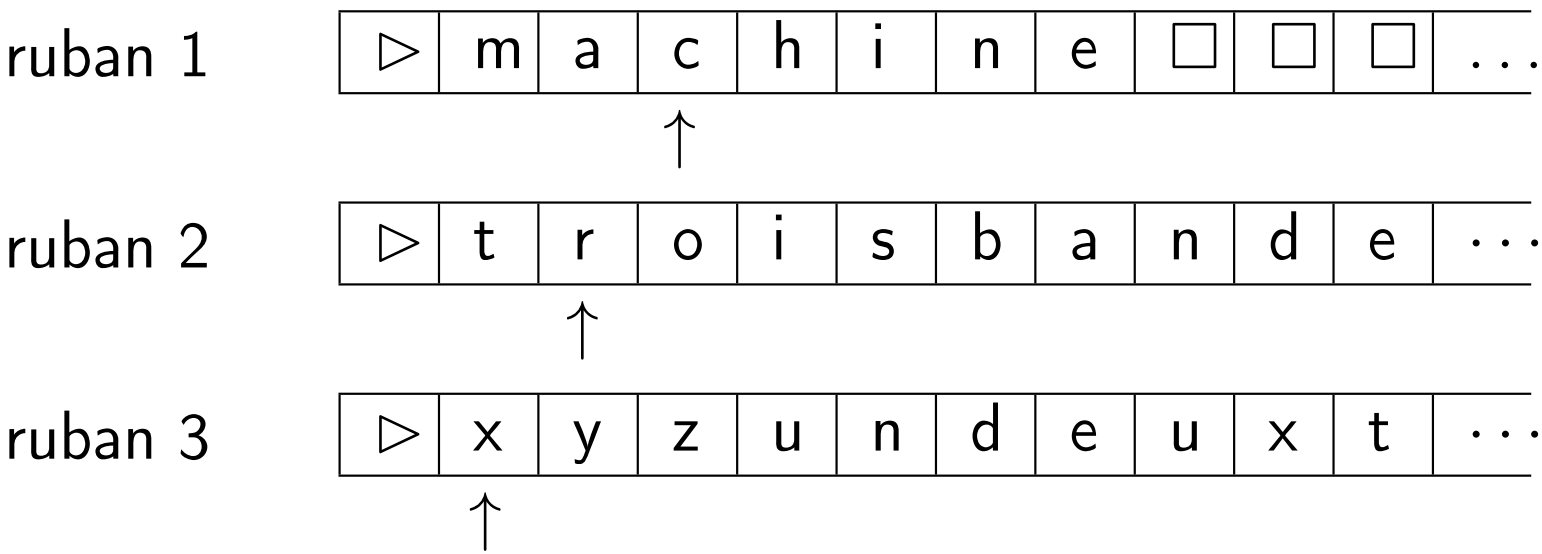
Lemme

*Si f est calculable par une **MT** $M = (\Sigma, Q, q_{start}, F, \delta)$ utilisant k rubans en temps T, alors f est calculable par une MT M' à un seul ruban en temps $5kT^2$.*

Nombre de rubans

Preuve

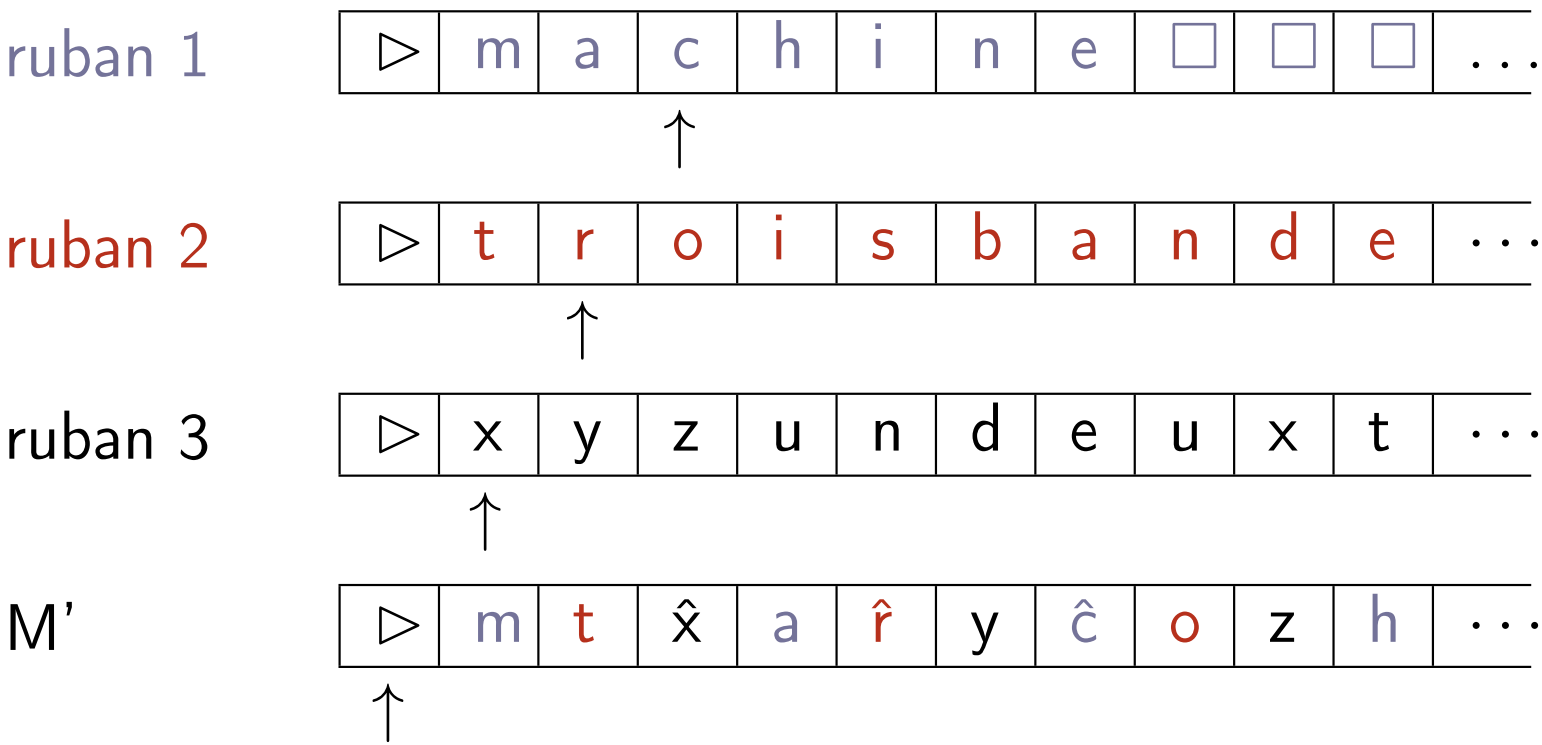
But : Simuler $M = (\Sigma, Q, q_{start}, F, \delta)$ utilisant k rubans par $M' = (\Sigma', Q', q'_{start}, F', \delta')$ MT avec un seul ruban



Nombre de rubans

Preuve

But : Simuler $M = (\Sigma, Q, q_{start}, F, \delta)$ utilisant k rubans par $M' = (\Sigma', Q', q'_{start}, F', \delta')$ MT avec un seul ruban



Nombre de rubans

Preuve (suite)

- Les cases i, $i + k$, $i + 2k$, ... du ruban de M' représentent le $i$ème ruban de M

Nombre de rubans

Preuve (suite)

- Les cases i, $i + k$, $i + 2k$, ... du ruban de M' représentent le $i$ème ruban de M
- Pour tout symbole $a \in \Sigma$: a, $\hat{a} \in \Sigma'$, où $\hat{}$ encode la position d'une tête de lecture de M.

Nombre de rubans

Preuve (suite)

- Les cases i, $i+k$, $i+2k$, ... du ruban de M' représentent le $i$ème ruban de M
- Pour tout symbole $a \in \Sigma$: a, $\hat{a} \in \Sigma'$, où $\hat{}$ encode la position d'une tête de lecture de M.
- Simuler 1 transition de M en cT **transitions de** M'
  - Parcours **gauche-droite** du ruban pour mémoriser (dans le registre de M') les cases pointées par les têtes de lecture de M
  - Parcours **droite-gauche** du ruban pour écrire et déplacer les têtes

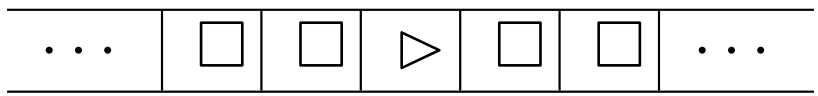
Nombre de rubans

Preuve (suite)

- Les cases i, $i+k$, $i+2k$, ... du ruban de M' représentent le $i$ème ruban de M
- Pour tout symbole $a \in \Sigma$: a, $\hat{a} \in \Sigma'$, où $\hat{}$ encode la position d'une tête de lecture de M.
- Simuler 1 transition de M en cT **transitions de** M'
  - Parcours **gauche-droite** du ruban pour mémoriser (dans le registre de M') les cases pointées par les têtes de lecture de M
  - Parcours **droite-gauche** du ruban pour écrire et déplacer les têtes
- M calcule f en temps T : au plus T cases écrites par M sur chacune de ces bandes $\Longrightarrow$ M' calcule f en $5kT^2$ transitions

Rubans bidirectionnels

Une machine est dites à **rubans bidirectionnels** si ses rubans sont infinis dans les deux directions.



Soit $f : \{0,1\}^* \to \{0,1\}^*$ et $T : \mathbb{N} \to \mathbb{N}$

Lemme

*Si f est calculable par une **MT** M utilisant des rubans bidirectionnels en temps T, alors f est calculable par une MT standard (à rubans unidirectionnels) en temps $4T$.*

Rubans bidirectionnels

Preuve

But : simuler une machine M à rubans bidirectionnels par une machine à rubans unidirectionnels.

Ruban de M :

$\cdots$	x	y	z	u	n	d	e	u	x	t	r	o	i	s	$\cdots$

Ruban de M' :

Rubans bidirectionnels

Preuve

But : simuler une machine M à rubans bidirectionnels par une machine à rubans unidirectionnels.

Ruban de M :

···	x	y	z	u	n	d	e	u	x	t	r	o	i	s	···

Ruban de M' :

▷	(e,u)	(d,x)	(n,t)	(u,r)	(z,o)	(y,i)	(x,s)	···

Machines de Turing universelles

Codage des MT

Rappel : une MT M à k rubans est définie par

- un alphabet Σ **fini**
- un ensemble d'états Q dont q_{start} et F **finis**
- une fonction $\delta : Q \times \Sigma^k \to Q \times \Sigma^k \times \{L, R, S\}^k$ **finie**

Codage des MT

Rappel : une MT M à k rubans est définie par

- un alphabet Σ **fini**
- un ensemble d'états Q dont q_{start} et F **finis**
- une fonction $\delta : Q \times \Sigma^k \to Q \times \Sigma^k \times \{L, R, S\}^k$ **finie**

Toute machine de Turing peut donc être encodée par un mot de $\{0, 1\}^*$

Codage des MT

Rappel : une MT M à k rubans est définie par

- un alphabet Σ **fini**
- un ensemble d'états Q dont q_{start} et F **finis**
- une fonction $\delta : Q \times \Sigma^k \to Q \times \Sigma^k \times \{L, R, S\}^k$ **finie**

Toute machine de Turing peut donc être encodée par un mot de $\{0, 1\}^*$

Exemple : programme C en mémoire d'un ordinateur

Codage des MT

Rappel : une MT M à k rubans est définie par

- un alphabet Σ **fini**
- un ensemble d'états Q dont q_{start} et F **finis**
- une fonction $\delta : Q \times \Sigma^k \to Q \times \Sigma^k \times \{L, R, S\}^k$ **finie**

Toute machine de Turing peut donc être encodée par un mot de $\{0,1\}^*$

Exemple : programme C en mémoire d'un ordinateur

Observation : il est donc possible de définir une MT dont **les entrées sont des MTs** (codées par des mots sur $\{0,1\}^*$)

Codage des MT

Rappel : une MT M à k rubans est définie par

- un alphabet Σ **fini**
- un ensemble d'états Q dont q_{start} et F **finis**
- une fonction $\delta : Q \times \Sigma^k \to Q \times \Sigma^k \times \{L, R, S\}^k$ **finie**

Toute machine de Turing peut donc être encodée par un mot de $\{0,1\}^*$

Exemple : programme C en mémoire d'un ordinateur

Observation : il est donc possible de définir une MT dont **les entrées sont des MTs** (codées par des mots sur $\{0,1\}^*$)

Exemple : un ordinateur

Codage des MT : exemple

Objectif : représenter M par un mot binaire **aisément décodable**

Exemple :

Codage des MT : exemple

Objectif : représenter M par un mot binaire **aisément décodable**

Exemple :

- $0^n 1 0^s 1 0^k 1 0^{f_1} 1 \cdots 1 0^{f_k} 1 0^m 1 0^u 1 \ldots$
  - n états, numérotés de 1 à n, dont s état initial, k la taille de $F = \{f_1, \ldots, f_k\}$
  - m symboles d'alphabet 1 à m, dont u le symbole blanc $\square$

Codage des MT : exemple

Objectif : représenter M par un mot binaire **aisément décodable**

Exemple :

- $0^n 1 0^s 1 0^k 1 0^{f_1} 1 \cdots 1 0^{f_k} 1 0^m 1 0^u 1 \ldots$
  - n états, numérotés de 1 à n, dont s état initial, k la taille de $F = \{f_1, \ldots, f_k\}$
  - m symboles d'alphabet 1 à m, dont u le symbole blanc $\square$
- puis les transitions : $\ldots 0^p 1 0^r 1 0^q 1 0^w 1 0^d 1 \ldots$

$$p, r \rightarrow q, w, d$$

Codage des MT : exemple

Objectif : représenter M par un mot binaire **aisément décodable**

Exemple :

- $0^n 10^s 10^k 10^{f_1} 1 \cdots 10^{f_k} 10^m 10^u 1 \ldots$
  - n états, numérotés de 1 à n, dont s état initial, k la taille de $F = \{f_1, \ldots, f_k\}$
  - m symboles d'alphabet 1 à m, dont u le symbole blanc $\square$
- puis les transitions : $\ldots 0^p 10^r 10^q 10^w 10^d 1 \ldots$

$$p, r \rightarrow q, w, d$$

- et enfin, un marqueur de fin $\ldots 111$ (au moins trois 1)

Codage des MT : exemple

Objectif : représenter M par un mot binaire **aisément décodable**

Exemple :

- $0^n 10^s 10^k 10^{f_1} 1 \cdots 10^{f_k} 10^m 10^u 1 \ldots$
  - n états, numérotés de 1 à n, dont s état initial, k la taille de $F = \{f_1, \ldots, f_k\}$
  - m symboles d'alphabet 1 à m, dont u le symbole blanc $\square$
- puis les transitions : $\ldots 0^p 10^r 10^q 10^w 10^d 1 \ldots$

$$p, r \rightarrow q, w, d$$

- et enfin, un marqueur de fin $\ldots 111$ (au moins trois 1)

Ex : langage machine `push 0x14` encodé par : `0x6A 0x14`

Codage des MT : propriétés

Dans la suite, on supposera l'existence d'un codage des MTs par des mots de $\{0,1\}^*$ tel que :

1. Tout mot de $\{0,1\}^*$ représente une MT

Codage des MT : propriétés

Dans la suite, on supposera l'existence d'un codage des MTs par des mots de $\{0,1\}^*$ tel que :

1. Tout mot de $\{0,1\}^*$ représente une MT

   Exemple : associer aux codes non valides une MT par défaut, par exemple une MT qui s'arrête immédiatement.

Codage des MT : propriétés

Dans la suite, on supposera l'existence d'un codage des MTs par des mots de $\{0,1\}^*$ tel que :

1. Tout mot de $\{0,1\}^*$ représente une MT

   Exemple : associer aux codes non valides une MT par défaut, par exemple une MT qui s'arrête immédiatement.

2. Toute machine de Turing M est codée par une infinité de mots de $\{0,1\}^*$

Codage des MT : propriétés

Dans la suite, on supposera l'existence d'un codage des MTs par des mots de $\{0,1\}^*$ tel que :

1. Tout mot de $\{0,1\}^*$ représente une MT

   Exemple : associer aux codes non valides une MT par défaut, par exemple une MT qui s'arrête immédiatement.

2. Toute machine de Turing M est codée par une infinité de mots de $\{0,1\}^*$

   Exemple : spécifier que le code se termine par un nombre arbitraire de 1 qui sont ignorés.

Codage des MT : propriétés

Dans la suite, on supposera l'existence d'un codage des MTs par des mots de $\{0,1\}^*$ tel que :

1. Tout mot de $\{0,1\}^*$ représente une MT

   Exemple : associer aux codes non valides une MT par défaut, par exemple une MT qui s'arrête immédiatement.

2. Toute machine de Turing M est codée par une infinité de mots de $\{0,1\}^*$

   Exemple : spécifier que le code se termine par un nombre arbitraire de 1 qui sont ignorés.

Notation :

- Pour $\alpha \in \{0,1\}^*$: M_α dénote la MT codée par α
- $\langle \mathrm{M} \rangle$ est une représentation de M sur $\{0,1\}^*$.

Machine de Turing universelle

Théorème

Il existe une Machine de Turing $\mathcal{U}$ telle que :

- *Pour tout $\alpha, x \in \{0,1\}^*$, $\mathcal{U}$ accepte (α, x) si et seulement si M_α accepte x (où M_α est la MT codée par le mot α)*

Machine de Turing universelle

Théorème

Il existe une Machine de Turing $\mathcal{U}$ telle que :

- *Pour tout $\alpha, x \in \{0,1\}^*$, $\mathcal{U}$ accepte (α, x) si et seulement si M_α accepte x (où M_α est la MT codée par le mot α)*

- *De plus, si M_α s'arrête sur l'entrée x **après** T **transitions**, alors $\mathcal{U}$ s'arrête sur (α, x) **après au plus** cT^2 **transitions** où c dépend de l'alphabet, du nombre de rubans, et du nombre d'états de M_α.*

Machine de Turing universelle

Théorème

Il existe une Machine de Turing $\mathcal{U}$ telle que :

- *Pour tout $\alpha, x \in \{0,1\}^*$, $\mathcal{U}$ accepte (α, x) si et seulement si M_α accepte x (où M_α est la MT codée par le mot α)*

- *De plus, si M_α s'arrête sur l'entrée x **après** T **transitions**, alors $\mathcal{U}$ s'arrête sur (α, x) **après au plus** cT^2 **transitions** où c dépend de l'alphabet, du nombre de rubans, et du nombre d'états de M_α.*

Remarques :

- il existe une version plus efficace en $cT \log T$ transitions
- diverses MT universelles ont été proposées, dont certaines ont moins de 10 états et moins de 10 symboles

Machine de Turing universelle

Preuve :

Trois rubans :

- Le 1er contient α, **la description de** M_α
- Le 2ème contient le mot d'entrée x, et est utilisé par $\mathcal{U}$ pour représenter **le contenu du ruban de** M_α
- Le 3ème sert à mémoriser **l'état** et la **position de la tête de lecture** de M_α

Machine de Turing universelle

Preuve :

Trois rubans :

- Le 1er contient α, **la description de** M_α
- Le 2ème contient le mot d'entrée x, et est utilisé par $\mathcal{U}$ pour représenter **le contenu du ruban de** M_α
- Le 3ème sert à mémoriser **l'état** et la **position de la tête de lecture** de M_α

À chaque étape, $\mathcal{U}$ **simule la prochaine transition de** M_α en utilisant le contenu des rubans, et elle les met à jour.

Questions

- Soit M une MT. Est-il possible d'écrire un programme en langage C (ou autre) qui simule M ?

Questions

- Soit M une MT. Est-il possible d'écrire un programme en langage C (ou autre) qui simule M ?

  cf. `https://turingmachinesimulator.com/`

Questions

- Soit M une MT. Est-il possible d'écrire un programme en langage C (ou autre) qui simule M ?

  cf. `https://turingmachinesimulator.com/`

- Étant donné un programme en langage C, est-il possible d'écrire une MT M qui simule ce programme ?

Questions

- Soit M une MT. Est-il possible d'écrire un programme en langage C (ou autre) qui simule M ?

  cf. `https://turingmachinesimulator.com/`

- Étant donné un programme en langage C, est-il possible d'écrire une MT M qui simule ce programme ?

  - programme C $\rightarrow$ programme ASM
  - coder la valeur des registres dans l'état de M
  - utiliser un ruban pour la pile, et un ruban pour le tas

Questions

- Soit M une MT. Est-il possible d'écrire un programme en langage C (ou autre) qui simule M ?

  cf. `https://turingmachinesimulator.com/`

- Étant donné un programme en langage C, est-il possible d'écrire une MT M qui simule ce programme ?

  - programme C $\rightarrow$ programme ASM
  - coder la valeur des registres dans l'état de M
  - utiliser un ruban pour la pile, et un ruban pour le tas

- Existe-il un programme C (ou autre) universel ?

Questions

- Soit M une MT. Est-il possible d'écrire un programme en langage C (ou autre) qui simule M ?

  cf. `https://turingmachinesimulator.com/`

- Étant donné un programme en langage C, est-il possible d'écrire une MT M qui simule ce programme ?

  - programme C $\rightarrow$ programme ASM
  - coder la valeur des registres dans l'état de M
  - utiliser un ruban pour la pile, et un ruban pour le tas

- Existe-il un programme C (ou autre) universel ?

  $\longrightarrow$ GDB, interpréteur Python, JVM, moteur JS d'un navigateur web, etc.

Thèse de Church-Turing

Les procédures de calcul effectives sont exactement celles exprimables par machine de Turing

Thèse de Church-Turing

Les procédures de calcul effectives sont exactement celles exprimables par machine de Turing

C'est une **thèse**, ni une hypothèse, ni un théorème

Thèse de Church-Turing

Les procédures de calcul effectives sont exactement celles exprimables par machine de Turing

C'est une **thèse**, ni une hypothèse, ni un théorème

Argument : les formalismes connus ont la **même expressivité**

- Machines de Turing (Alan Turing 1936)
- Systèmes de réécriture de Post (Emil Post 1936)
- Fonctions récursives (Jacques Herbrand, Kurt Gödel 1931-34)
- λ-calcul (Alonzo Church 1933, Stephen C. Kleene 1935)
- Logique combinatoire (Moses Schönfinkel 1924, Haskell B. Curry 1929)

Thèse de Church-Turing forte

Toute procédure de calcul effective est simulable par une MT avec un surcoût polynomial.

C'est à dire : T pas de calcul de la procédure sont simulés par T^c transitions de la machine de Turing où c est une constante qui dépend uniquement de la procédure (pas du mot d'entrée).

Thèse de Church-Turing forte

Toute procédure de calcul effective est simulable par une MT avec un surcoût polynomial.

C'est à dire : T pas de calcul de la procédure sont simulés par T^c transitions de la machine de Turing où c est une constante qui dépend uniquement de la procédure (pas du mot d'entrée).

Controversée : ordinateurs quantiques

Machines de Turing non-déterministes

MT non-déterministe

Généralisation des machines de Turing :

$$\delta \subseteq Q \times \Sigma^k \times Q \times \Sigma^k \times \{L, R, S\}^k$$

est une **relation** (et non plus une fonction)

Exemple : "Déterminer si l'avant-dernier caractère de $w \in \{0,1\}^*$ est un 0"

$q_{start}, 1 \to q_{start}, 1, R$
$q_{start}, 0 \to q_{start}, 0, R$
$q_{start}, 0 \to q_{last}, 0, R$

$q_{last}, 0 \to q_{\square}, 0, R$
$q_{last}, 1 \to q_{\square}, 1, R$
$q_{\square}, \square \to q_{yes}, \square, R$

Exécution sur $w = 10101$?

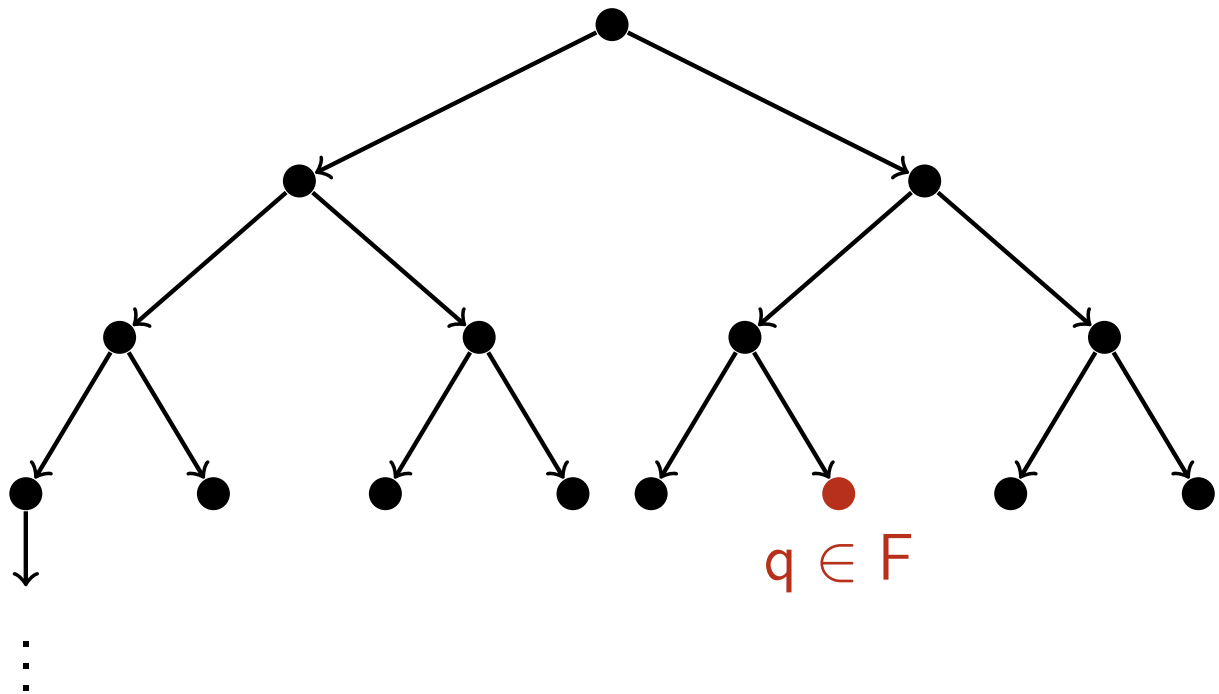
Calcul non-déterministe

L'exécution d'une M **non-déterministe** sur un mot w produit un **arbre**.

Calcul non-déterministe

L'exécution d'une M **non-déterministe** sur un mot w produit un **arbre**.

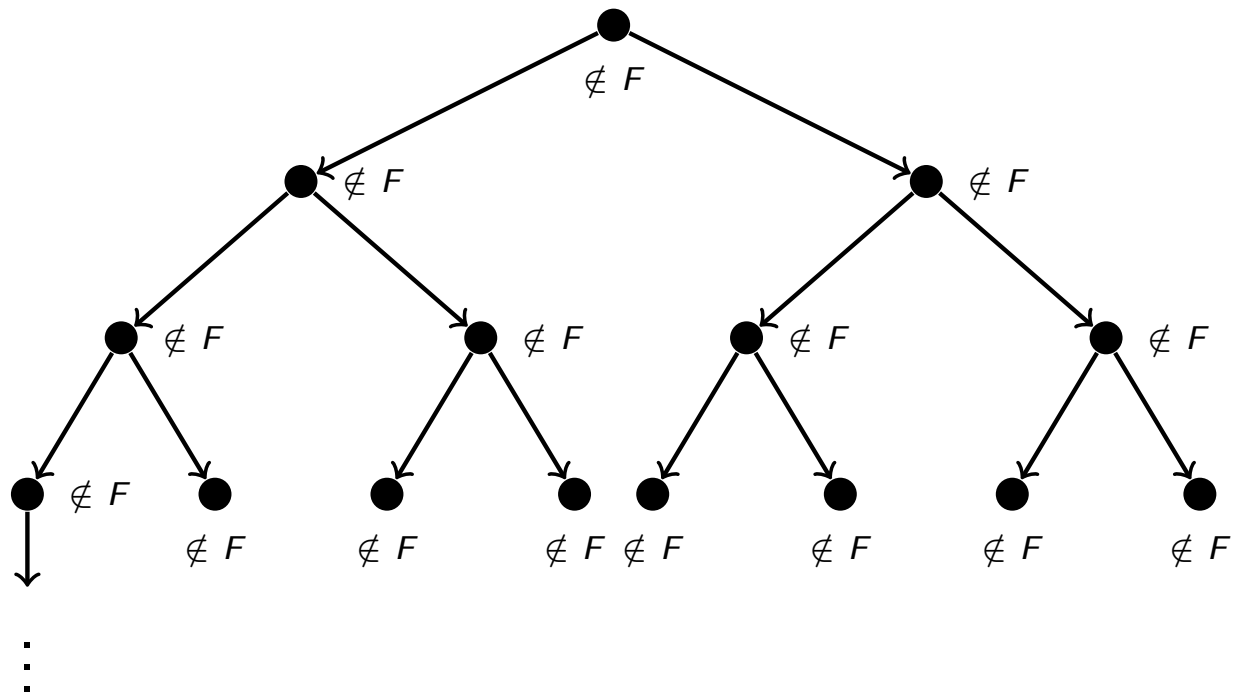
- M accepte w si (au moins) **une** branche de l'arbre aboutit dans un état acceptant $q \in F$



Calcul non-déterministe

L'exécution d'une M **non-déterministe** sur un mot w produit un **arbre**.

- M accepte w si (au moins) **une** branche de l'arbre aboutit dans un état acceptant $q \in F$
- M n'accepte pas w si **aucune** branche de l'arbre aboutit dans un état acceptant $q \in F$

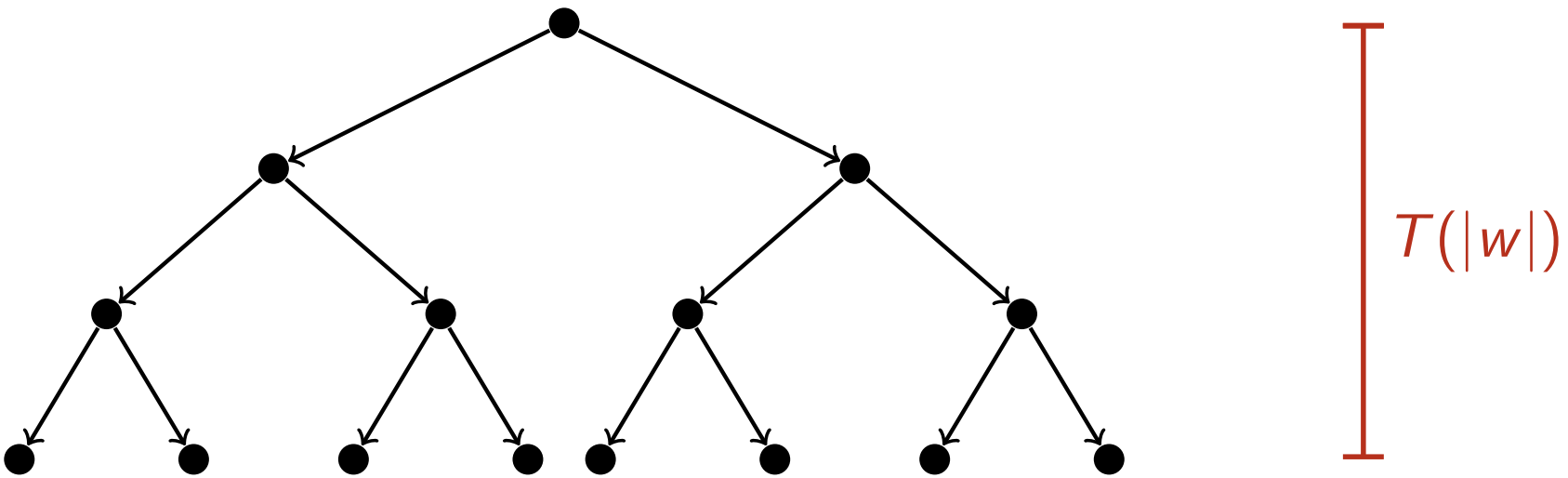


Temps de calcul non-déterministe

Soit M une MT non-déterministe **qui s'arrête pour tout mot d'entrée**, et $T : \mathbb{N} \to \mathbb{N}$ une fonction.

Définition

M s'exécute en temps T si pour tout mot $w \in \{0,1\}^*$, l'arbre d'exécution de M sur w est de hauteur au plus $T(|w|)$.



Expressivité du non-déterminisme

Existe-t-il des langage décidés par une MT **non-déterministe**, mais décidés par aucune MT déterministe ?

Soit $f : \{0,1\}^* \to \{0,1\}^*$ et $T : \mathbb{N} \to \mathbb{N}$

Lemme

*Si f est calculable par une **MT** M non-déterministe en temps T, alors f est calculable par une MT déterministe en temps $2^{\mathcal{O}(T)}$.*

Expressivité du non-déterminisme

Preuve

- L'exécution d'une machine de Turing non-déterministe est un **arbre** A

Expressivité du non-déterminisme

Preuve

- L'exécution d'une machine de Turing non-déterministe est un **arbre** A
- On construit une machine déterministe M' qui effectue une recherche **en largeur d'abord** de A

Expressivité du non-déterminisme

Preuve

- L'exécution d'une machine de Turing non-déterministe est un **arbre** A
- On construit une machine déterministe M' qui effectue une recherche **en largeur d'abord** de A
- Initialement, M' place la configuration initiale de M sur son premier ruban

Expressivité du non-déterminisme

Preuve

- L'exécution d'une machine de Turing non-déterministe est un **arbre** A
- On construit une machine déterministe M' qui effectue une recherche **en largeur d'abord** de A
- Initialement, M' place la configuration initiale de M sur son premier ruban
- À chaque itération, le premier ruban contient une liste de configurations (un niveau de l'arbre A)

Expressivité du non-déterminisme

Preuve

- L'exécution d'une machine de Turing non-déterministe est un **arbre** A
- On construit une machine déterministe M' qui effectue une recherche **en largeur d'abord** de A
- Initialement, M' place la configuration initiale de M sur son premier ruban
- À chaque itération, le premier ruban contient une liste de configurations (un niveau de l'arbre A)
- M' calcule les configurations suivantes sur son 2ème ruban en simulant M (le niveau suivant dans A). M' utilise d'autres rubans pour la simulation.

Expressivité du non-déterminisme

Preuve

- L'exécution d'une machine de Turing non-déterministe est un **arbre** A
- On construit une machine déterministe M' qui effectue une recherche **en largeur d'abord** de A
- Initialement, M' place la configuration initiale de M sur son premier ruban
- À chaque itération, le premier ruban contient une liste de configurations (un niveau de l'arbre A)
- M' calcule les configurations suivantes sur son 2ème ruban en simulant M (le niveau suivant dans A). M' utilise d'autres rubans pour la simulation.
- Puis M' recopie son 2ème ruban sur le premier et procède à l'itération suivante