

Modèles de Calcul

Yassine Lakhnech

Yassine.Lakhnech@imag.fr

2007/08

Université Joseph Fourier

Lab.: VERIMAG

Équipe pédagogique

- Cours : Saddek Bensalem et Yassine Lakhnech
- Travux dirigés : Ph. Bidinger et Ph. Bizard

web: www-verimag.imag.fr/~lakhnech/MCAL

Bibliographie

- J. Hopcroft, R. Motwani, J. Ullman, Introduction to Automata Theory, Languages and Computation, 2nd edition, Addison-Wesley, 2001
- P. Wolper. Introduction à la calculabilité - 2ième édition, Dunod, 2001.
- Cl.Benzaken, Systèmes Formels, Masson, 1991.
- Transparents d'INF232
<http://www-verimag.imag.fr/~lakhnech>

Motivation

Comprendre les limites de l'informatique.

- Que veut dire qu'une fonction soit *calculable* ou qu'un problème soit *soluble* par des algorithmes.
- Existe-il des problèmes *insolubles* par des algorithmes.
- Peut-on avoir des réponses à ces questions *indépendantes* :
 - du langage de programmation
 - de l'ordinateur sur lequel les programmes sont exécutés

Plusieurs modèles de calcul

- Mémoire finie : Automates d'états finis, expressions régulières.
- Mémoire finie + pile : Automates à pile
- Mémoire infinie :
 - Machines de Turing (Alan Turing)
 - Systèmes de Post (Emil Post)
 - Fonctions μ -récurives (Kurt Gödel, Jacques Herbrand)
 - λ -Calcul (Alonzo Church, Stephen C. Kleene)
 - Logique des combinateurs (Moses Schönfinkel, Haskell B. Curry)

Problèmes et Langages

- Quels problèmes sont solubles par un programme exécuté sur un ordinateur?

Il faut préciser :

- la notion de problème
 - Fonction de $\mathbb{N}^k \rightarrow \mathbb{N}$ ou
 - **Langage :**
 - **Alphabet :** les symboles pour décrire les instances du problème
 - **Le langage** qui décrit les instances positives
- la notion de programme exécuté sur un ordinateur :
Machine de Turing

Exemples de Problèmes

- - **Entrée** : le codage binaire $\hat{n}$ d'un entier naturel $n \in \mathbb{N}$
 - **Sortie** : n est pair

Formalisation

- - $\Sigma = \{0, 1\}$
 - $P = \{u \in \Sigma^* \mid \exists n \in \mathbb{N} \cdot \hat{n} = u \wedge n \equiv_2 0\}$.
- Il existe un programme avec une mémoire fini pour résoudre ce problème : P est un langage régulier.

Exemples de Problèmes

- - **Entrée** : le codage binaire $\hat{n}$ d'un entier naturel $n \in \mathbb{N}$
 - **Sortie** : n est un nombre premier

Formalisation

- $\Sigma = \{0, 1\}$
- $P = \{u \in \Sigma^* \mid \exists n \in \mathbb{N} \cdot \hat{n} = u \wedge n \text{ premier}\}.$
- Il n'existe pas de programme avec une mémoire fini pour résoudre ce problème : P n'est pas régulier.
- Par contre, il existe un programme avec une mémoire infinie pour résoudre ce problème.

Exemples de Problèmes

- - **Entrée** : une chaîne de caractères $\hat{\pi}$ représentant un C-programme π
 - **Sortie** : π s'arrête sur l'entrée 0

Formalisation

- Σ l'alphabet du langage C
- $P = \{ \pi \in \Sigma^* \mid \pi \text{ est un programme } C \text{ et l'exécution de } \pi \text{ sur } 0 \text{ s'arrête} \}$
- Il n'existe pas de programme même avec une mémoire infinie pour résoudre ce problème.

Plan du cours

- Machine de Turing
- Langage décidable
- Langage indécidable
- Langage récursivement énumérable (semi-décidable)
- Théorème de Rice

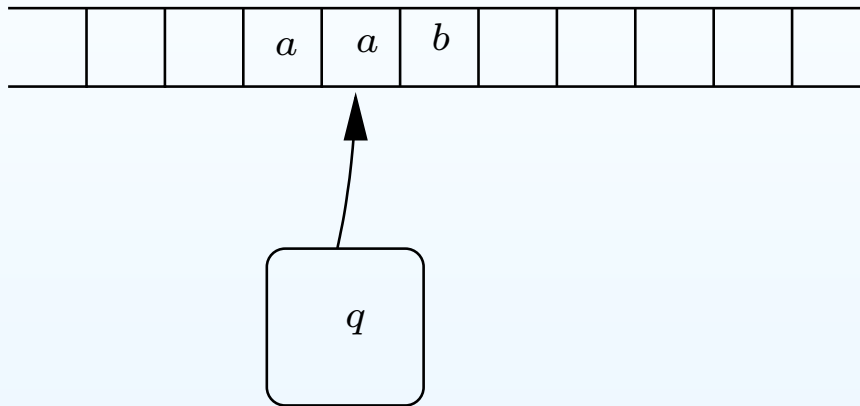
Machines de Turing

Alan Turing inventa cette machine abstraite en 1936 pour définir la notion de "fonction calculable".

- toute tâche exécutée par une machine de Turing MT peut l'être sur un ordinateur ... et vice-versa !
- peut simuler n'importe quel automate fini, automates à pile et même n'importe quel programme exécutable sur un ordinateur
- il existe une MT universelle capable de simuler toutes les autres MT : programme comme donnée

Machines de Turing

- Une MT est composée d'une unité de contrôle, d'un ruban infini divisé en cases et d'une tête de lecture/ écriture.



- en fonction de ce qu'elle lit sur la case courante et de son état courant (conformément à sa relation de transition δ), elle
 1. écrit un symbole sur sa case courante
 2. déplace la tête de lecture/ écriture d'une case (à droite, à gauche)
 3. change d'état.

Machines de Turing

Une *Machine de Turing* M est donnée par

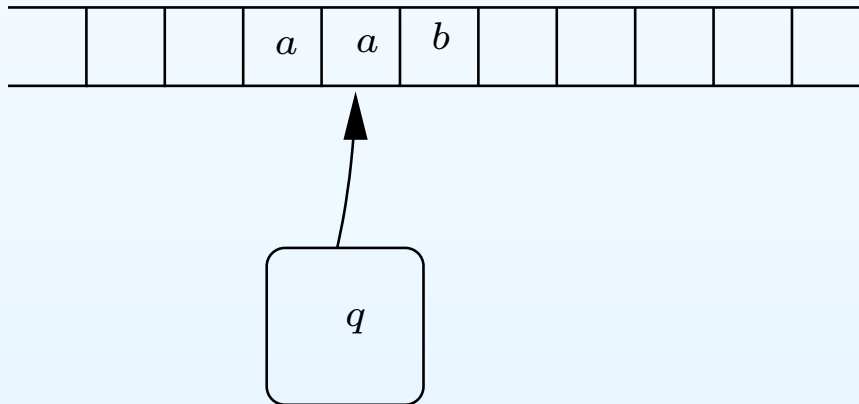
$(Q, \Sigma, \Gamma, \delta, q_0, F)$ où :

- Q est un ensemble fini d'états.
- Σ est l'alphabet (fini) d'entrée
- Γ est l'alphabet (fini) du ruban, tel que $\Sigma \subseteq \Gamma$
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{G, D, I\}$ est la fonction de transition.
- q_0 est l'état initial.
- $F \subseteq Q$ est l'ensemble des états accepteurs.
- On suppose qu'aucune transition n'est définie dans les états accepteurs.
- On considère un symbole $B \in \Gamma \setminus \Sigma$ particulier dit **blanc**.

Configuration

Une *configuration* de P est un élément $(u, q, v) \in \Gamma^* \times \Sigma \times \Gamma^*$ où

- q est l'état courant.
- u est le mot qui est à gauche de la tête de lecture/ écriture.
- v est le mot qui est à droite de la tête de lecture/ écriture.



est représentée par (a, q, ab)

Relation de Transition

La machine M réalise une transition d'une configuration (u, q, v) vers une configuration (u', q', v') , noté

$$(u, q, v) \rightarrow (u', q', v'),$$

si une des conditions suivantes est satisfaite :

- $v = av'', \delta(q, a) = (q', b, G)$ **et** $u = u'c, v' = cbv''$
- $v = av'', \delta(q, a) = (q', b, d)$ **et** $u' = ub, v' = v''$
- $\delta(q, a) = (q', b, I)$ **et** $u' = u, v' = bv''$

Langage reconnu

- Une configuration initiale est de la forme (ϵ, q_0, u) .
- Une configuration finale est de la forme (u, q, v) avec $q \in F$.
- Un mot u est accepté par M , s'il existe $q \in F$ avec :

$$(\epsilon, q_0, u) \rightarrow^* (u', q, v').$$

- Le langage reconnu par M est

$$L(M) = \{u \mid \exists q \in F \cdot (\epsilon, q_0, u) \rightarrow^* (u', q, v')\}.$$

Exemple

Le langage $L = \{a^i b^i \mid i \geq 0\}$ est reconnu par la Machine de Turing définie par $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ où :

- $Q = \{q_0, q_1, q_2, q_3, q_4\}$.
- $\Sigma = \{a, b\}$.
- $\Gamma = \{a, b, X, Y, B\}$.
- - $\delta(q_0, a) = (q_1, X, D) \quad \delta(q_0, b) = (q_2, Y, G) \quad \delta(q_1, a) = (q_1, a, D)$
 - $\delta(q_1, b) = (q_2, Y, G) \quad \delta(q_1, Y) = (q_1, Y, D) \quad \delta(q_2, a) = (q_2, a, G)$
 - $\delta(q_2, X) = (q_0, X, D) \quad \delta(q_2, Y) = (q_2, Y, G) \quad \delta(q_3, Y) = (q_3, Y, D)$
 - $\delta(q_3, B) = (q_4, B, D)$
- q_0 est l'état initial.
- $F = \{q_4\}$

Exemple (2)

Le langage $\{ a^n b^n c^n : n \geq 1 \}$ est reconnu par la Machine de Turing définie par $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ où :

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$, $\Sigma = \{a, b, c\}$,

$\Gamma = \{a, b, c, X, Y, Z, B\}$, $F = \{q_6\}$, et δ est défini par

$\delta(q_0, a) = (q_1, X, D)$	$\delta(q_1, a) = (q_1, a, D)$	$\delta(q_1, y) = (q_1, Y, D)$
$\delta(q_1, b) = (q_2, Y, D)$	$\delta(q_2, b) = (q_2, b, D)$	$\delta(q_2, z) = (q_2, Z, D)$
$\delta(q_2, c) = (q_3, Z, G)$	$\delta(q_3, z) = (q_3, Z, G)$	$\delta(q_3, b) = (q_3, b, G)$
$\delta(q_3, y) = (q_3, Y, G)$	$\delta(q_3, a) = (q_3, a, G)$	$\delta(q_3, x) = (q_0, X, D)$
$\delta(q_0, y) = (q_4, Y, D)$	$\delta(q_4, y) = (q_4, Y, D)$	$\delta(q_4, z) = (q_5, Z, D)$
$\delta(q_5, z) = (q_5, Z, D)$	$\delta(q_5, B) = (q_6, B, D)$	

Exécution

Une exécution de la machine, qui permet d'accepter le mot *aabbcc*,:

$$\begin{aligned} q_0 a a b b c c &\rightarrow x q_1 a b b c c \rightarrow x a q_1 b b c c \rightarrow x a y q_2 b c c \rightarrow x a y b q_2 c c \rightarrow \\ &x a y q_3 b z c \rightarrow \\ &x a q_3 y b z c \rightarrow x q_3 a y b z c \rightarrow q_3 x a y b z c \rightarrow x q_0 a y b z c \rightarrow x x q_1 y b z c \rightarrow \\ &x x y q_1 b x c \rightarrow \\ &x x y y q_2 z c \rightarrow x x y y z q_2 c \rightarrow x x y y q_3 z z \rightarrow x x y q_3 y z z \rightarrow x x q_3 y y z z \rightarrow \\ &x q_3 x y y z z \rightarrow \\ &x x q_0 y y z z \rightarrow x x y q_4 y z z \rightarrow x x y y q_4 z z \rightarrow x x y y z q_5 z \rightarrow \\ &x x y y z z q_5 B \rightarrow x x y y z z B q_6 B. \end{aligned}$$

Un langage de programmation simple

On présente un langage de programmation simple qui nous permet de décrire des machines de Turing de manière plus structurée sous forme de programme impératif.

$$\text{bexp} ::= R = S \mid \neg \text{bexp} \mid \text{bexp}_1 \wedge \text{bexp}_2 \mid \text{bexp}_1 \vee \text{bexp}_2$$

bexp expression booléenne, R la seule variable, $S \in \Gamma$ un symbole de l'alphabet du ruban.

$$P ::= R := S \mid D \mid G \mid P_1; P_2 \mid \text{bexp} \triangleright P \mid (\text{bexp} \triangleright P)^* \\ P_1 \parallel P_2 \mid \text{STOP} \mid \checkmark$$

Sémantique

Configuration : (u, P, v) .

On définit la relation de transition

$$(u, P, v) \rightarrow (u', P, v').$$

- $(u, R := S, S'v) \rightarrow (u, \surd, Sv')$
- $(u, D, Sv) \rightarrow (uS, \surd, v)$
- $(uS, G, v) \rightarrow (u, \surd, Sv)$
- $$\frac{(u, P_1, v) \rightarrow (u', P'_1, v') \quad P'_1 \neq \surd, \text{STOP}}{(u, P_1; P_2, v) \rightarrow (u', P'_1; P_2, v')}$$
- $$\frac{(u, P_1, v) \rightarrow (u', P'_1, v') \quad P'_1 = \surd}{(u, P_1; P_2, v) \rightarrow (u', P_2, v')}$$
- $$\frac{(u, P_1, v) \rightarrow (u', P'_1, v') \quad P'_1 = \text{STOP}}{(u, P_1; P_2, v) \rightarrow (u', \text{STOP}, v')}$$

Sémantique - II

- $(u, \text{bexp} \triangleright P, Sv) \rightarrow (u, P, Sv)$, si $\text{bexp}[S/R]$ est vrai
- $(u, (\text{bexp} \triangleright P)^*, Sv) \rightarrow (u, P; (\text{bexp} \triangleright P)^*, Sv)$, si $\text{bexp}[S/R]$ est vrai
- $(u, (\text{bexp} \triangleright P)^*, Sv) \rightarrow (u, \surd, Sv)$, si $\text{bexp}[S/R]$ n'est pas vrai
- $(u, P_1 \parallel P_2, v) \rightarrow (u, P_1, v)$
- $(u, P_1 \parallel P_2, v) \rightarrow (u, P_2, v)$

Un mot u est accepté par P , si

$$(\epsilon, P, u) \rightarrow^* (u', \surd, v')$$

Le langage $L(P)$ reconnu par une programme P est l'ensemble des mots acceptés par ce programme.

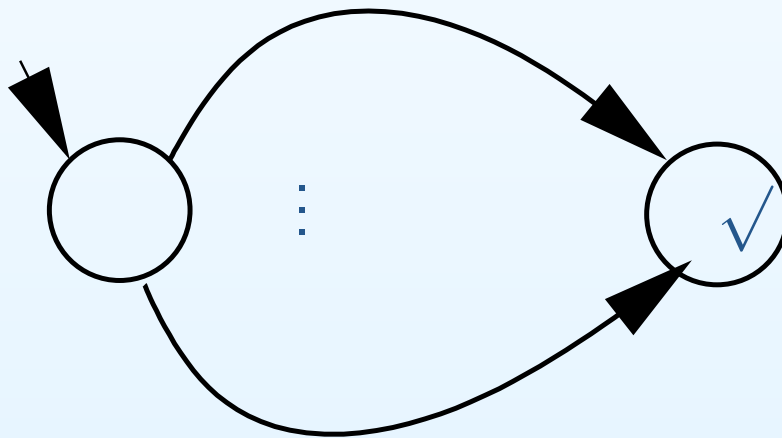
Des programmes vers les MT

Nous présentons sans trop donner de détail une fonction qui associe à chaque programme P de notre langage simple une MT $T(P)$ qui reconnaît le même langage.

Soit $\Gamma = \{a_1, \dots, a_n\}$.

$T(R := S)$ est la machine

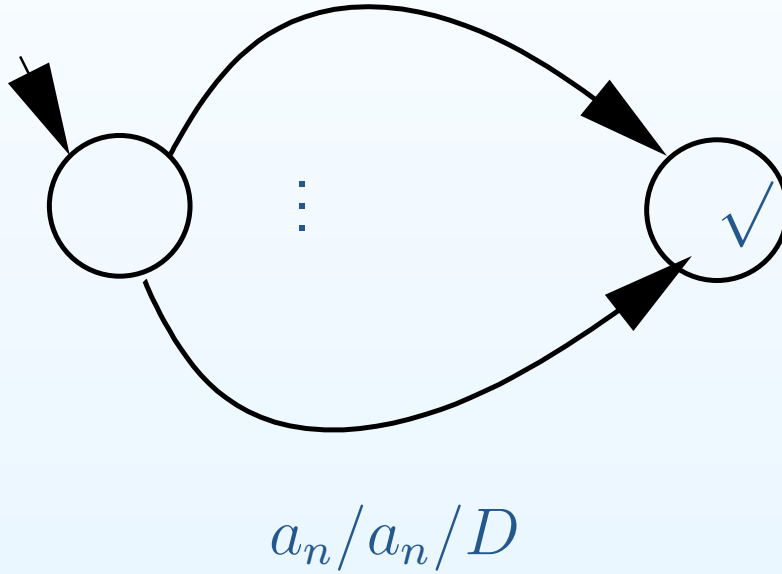
$a_1/S/I$



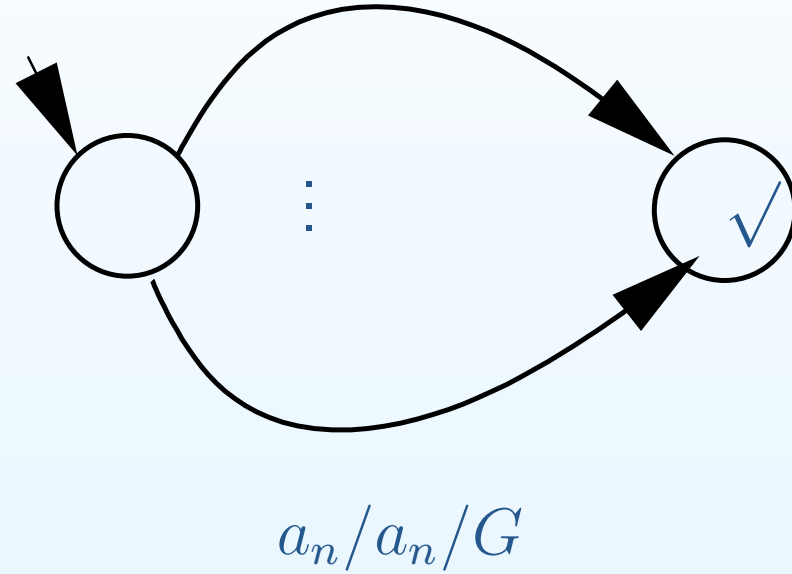
$a_n/S/I$

Des programmes vers les MT - II

$T(D)$ est la machine
 $a_1/a_1/D$

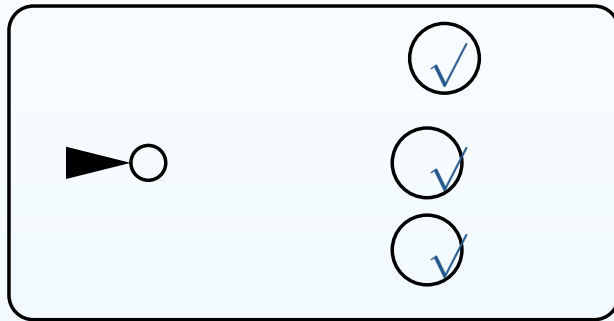


$T(G)$ est la machine
 $a_1/a_1/G$

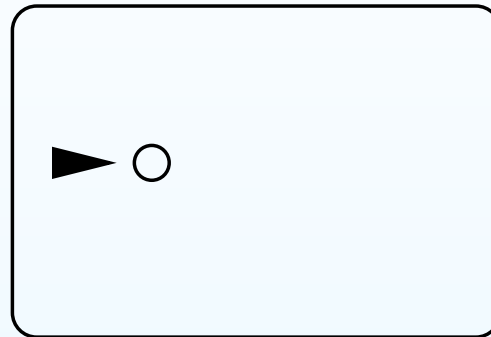


Des programmes vers les MT - II

$T(P_1; P_2)$ est la composition des machine $T(P_1)$ et $T(P_2)$



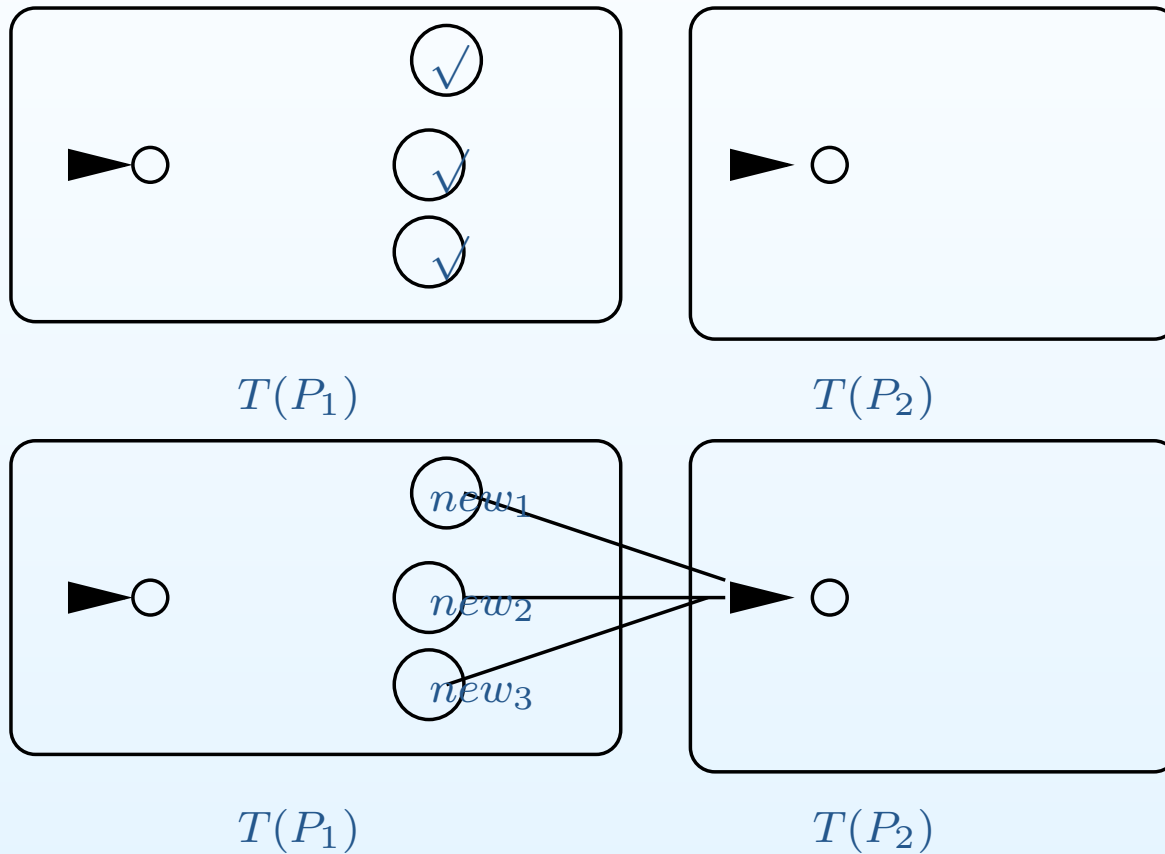
$T(P_1)$



$T(P_2)$

Des programmes vers les MT - II

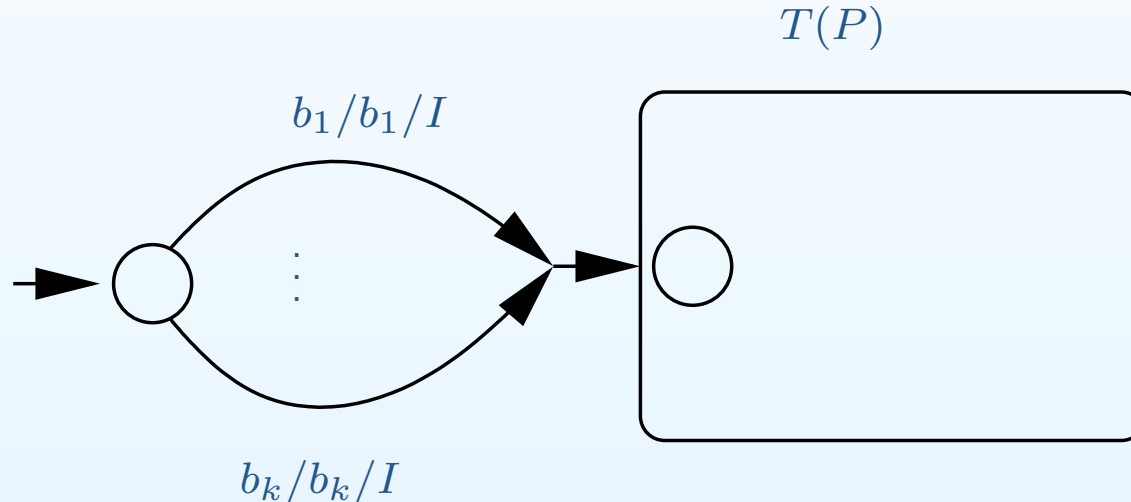
$T(P_1; P_2)$ est la composition des machine $T(P_1)$ et $T(P_2)$



Des programmes vers les MT - III

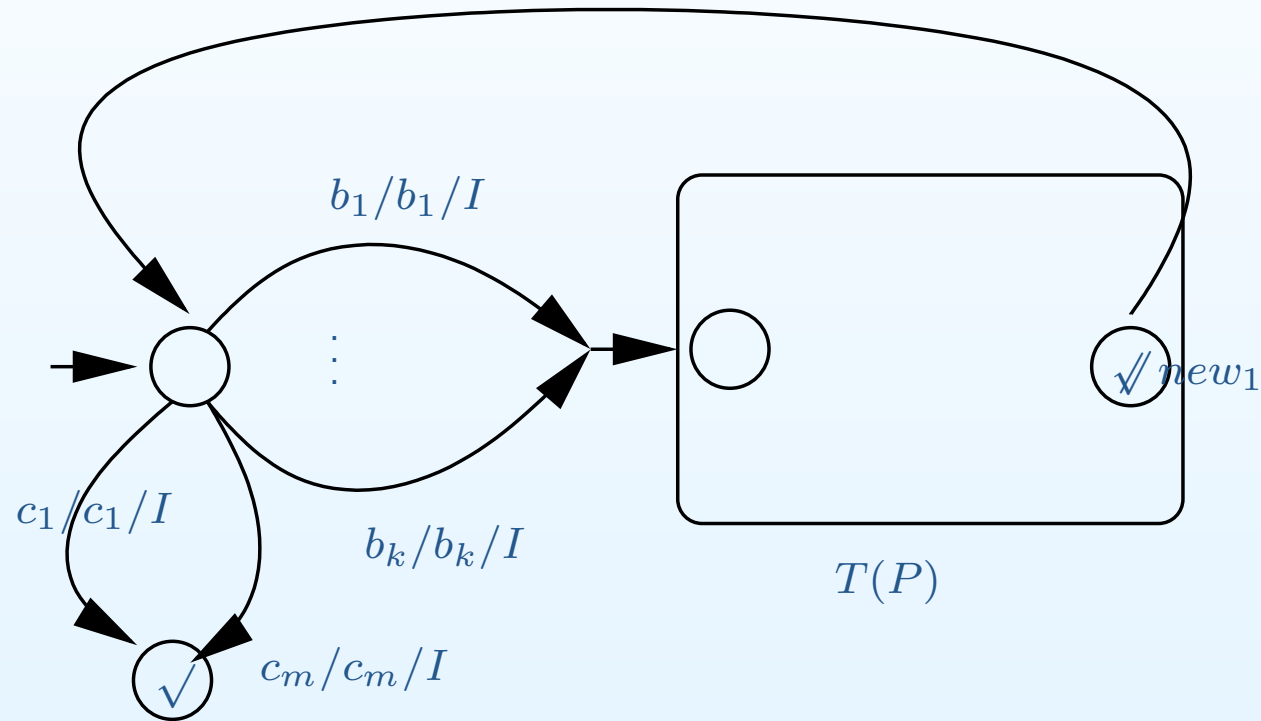
Etant une expression booléenne bexp , soit $S(\text{bexp})$ l'ensemble des symboles S de Γ qui tels que $\text{bexp}[S/R]$ est vrai.

Alors, $T(\text{bexp} \triangleright P)$ est la machine suivante où $S(\text{bexp}) = \{b_1, \dots, b_k\}$.



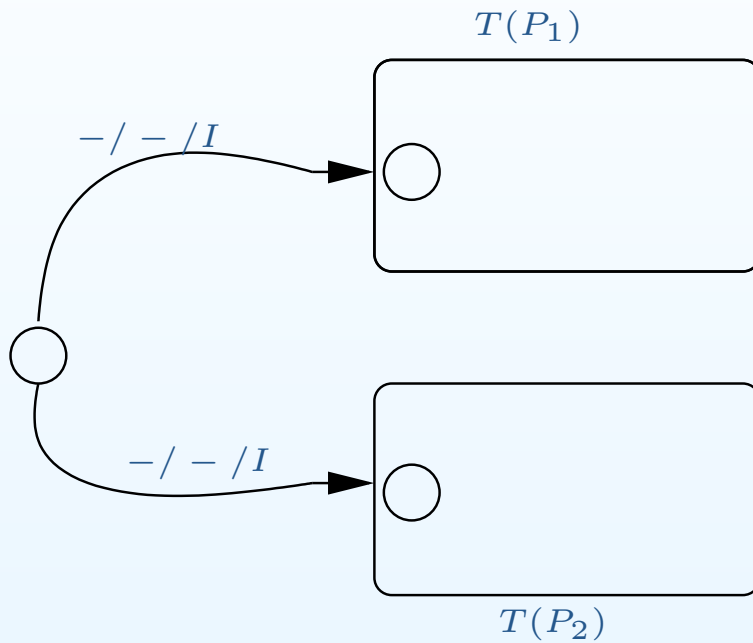
Des programmes vers les MT - V

$T((\text{bexp} \triangleright P)^*)$ est la machine suivaye où $S(\text{bexp}) = \{b_1, \dots, b_k\}$
 et $\Gamma \setminus \{b_1, \dots, b_k\} = \{c_1, \dots, c_m\}..$
 $-/-/I$

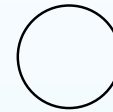


Des programmes vers les MT - VI

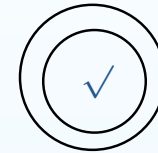
$T(P_1 \parallel P_2)$ est la machine



Traductions de STOP et $\checkmark$.



STOP



$\checkmark$

Les sorties possibles d'une machine de Turing

Soit M une machine de Turing.

On dit que M s'arrête pour l'entrée $u \in \Sigma^*$, s'il existe une configuration (v, q, w) telle que $(\epsilon, q_0, u) \rightarrow^* (v, q, aw)$ et $\delta(q, a)$ n'est pas défini.

C.a.d. M atteint une configuration où aucune transition n'est possible.

Pour une entrée u , M peut :

1. s'arrêter dans un état accepteur
2. s'arrêter dans un état qui n'est pas accepteur
3. ne jamais s'arrêter.

Langage récursif

Definition Un langage $L \subseteq \Sigma^*$ est *récursif*, s'il existe une machine de Turing M telle que :

1. $L(M) = L$ et
2. M s'arrête pour tout mot dans Σ^* .



On dit que L est *décidé* par M .

Langage récursivement énumérable

Definition Un langage $L \subseteq \Sigma^*$ est *récursivement énumérable (r.e.)*, s'il existe une machine de Turing M telle que $L(M) = L$.

□

Autrement dit, L est reconnaissable par une machine de Turing.

Liens entre récursifs et récursivement énumérables

Théorème

- Un langage L est décidable ssi son complément L^c est décidable.
- Un langage L est décidable ssi L et son complément L^c sont récursivement énumérable.

Liens entre récursifs et récursivement énumérables

Théorème

- Un langage L est décidable ssi son complément L^c est décidable.
- Un langage L est décidable ssi L et son complément L^c sont récursivement énumérable.

Preuve

1. Si L est décidé par une MT M alors L^c est décidé par la MT M' obtenue à partir de M en inversant les états accépteurs et non-accépteurs.



Liens entre récursifs et récursivement énumérables

Théorème

- Un langage L est décidable ssi son complément L^c est décidable.
- Un langage L est décidable ssi L et son complément L^c sont récursivement énumérable.

Preuve

1. Si L est décidé par une MT M alors L^c est décidé par la MT M' obtenue à partir de M en inversant les états accépteurs et non-accépteurs.
2. Si L est décidé par une MT M alors L est reconnu par M et L^c est reconnu par M' .



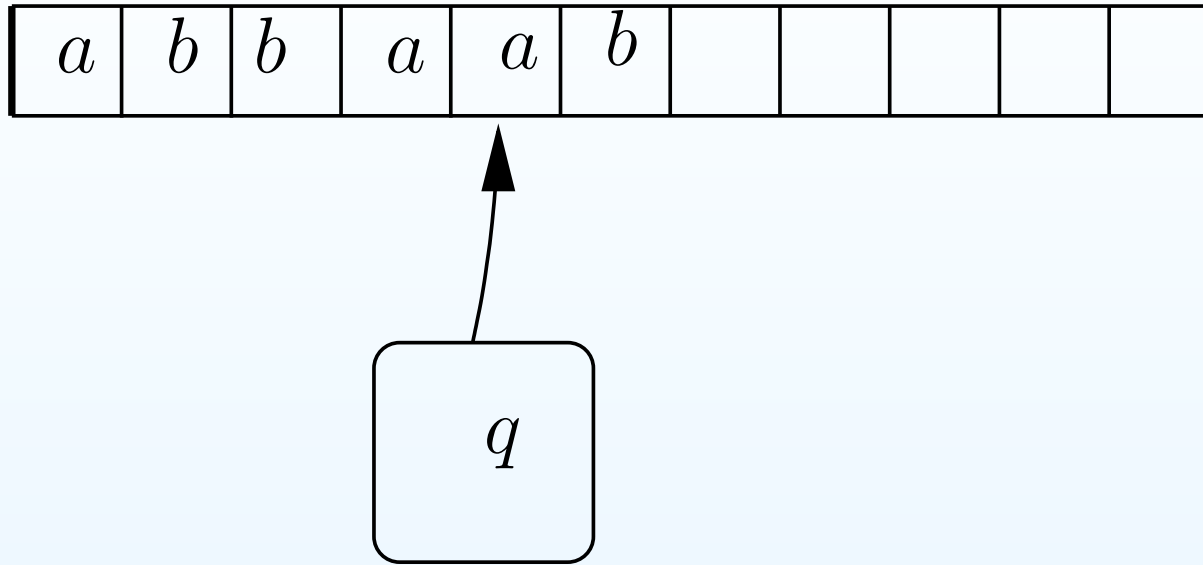
Machine de Turing à plusieurs rubans

- Une MT à k rubans, avec $k \geq 1$, est composée d'une unité de contrôle, de k rubans infinis divisé en cases, de k têtes de lecture/ écriture.
Chaque rubans a sa propre tête de lecture.
- en fonction de ce qu'elle lit sur les k cases courantes et de son état courant (conformément à sa relation de transition δ), elle
 1. écrit un symbole sur la case courante de chaque ruban
 2. déplace chacune des têtes de lecture/ écriture d'une case (à droite, à gauche)
 3. change d'état.

Equivalence entre Machines à 1-ruban et plusieurs rubans

Théorème Tout langage reconnu (décidié) par une machine de Turing à plusieurs rubans est récursivement énumérable (décidable).

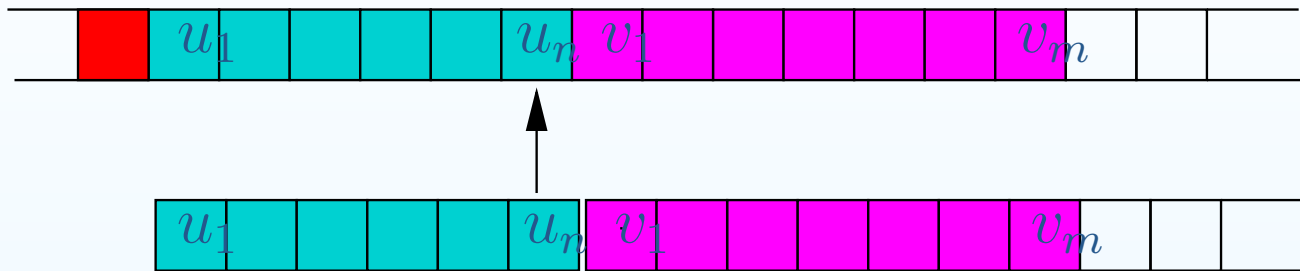
Machine de Turing à semi-ruban



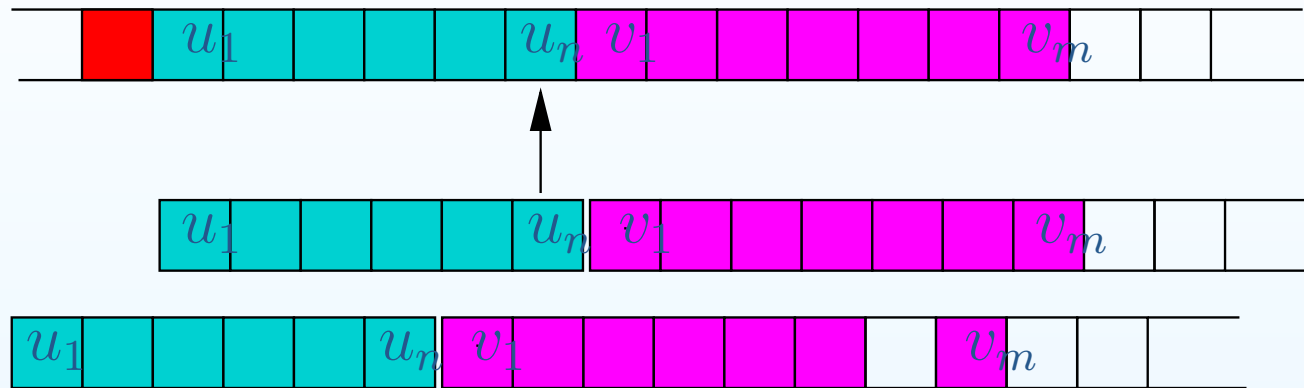
Equivalence entre Machine de Turing et Machine de Turing à semi-ruban

Théorème Tout langage récursivement énumérable (décidable) est reconnaissable (décidable) par une machine de Turing à semi-ruban.

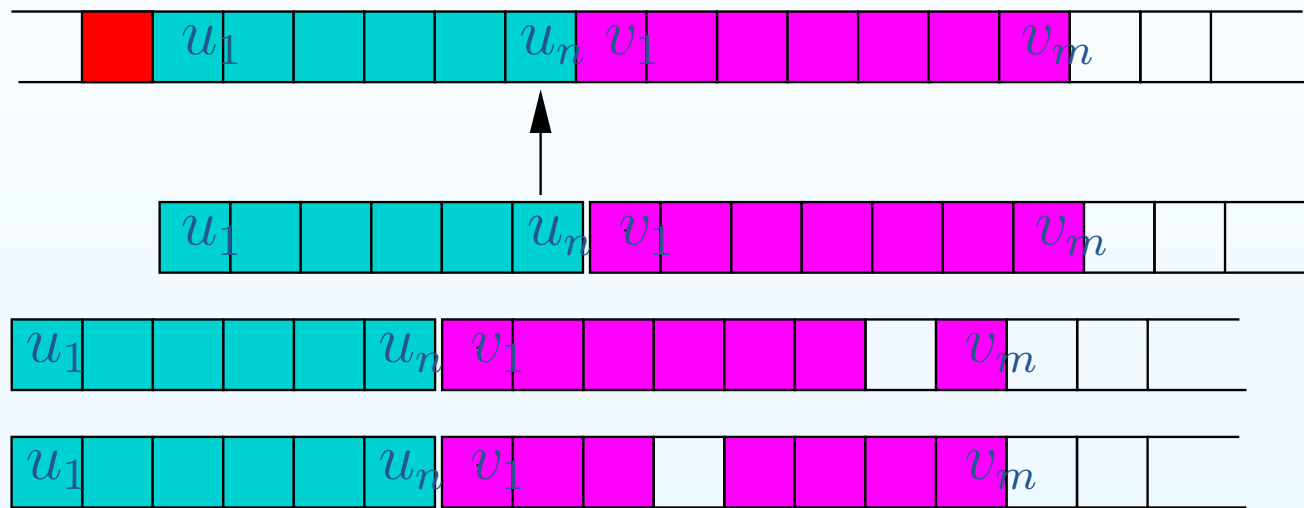
Equivalence entre Machine de Turing et Machine de Turing à semi-ruban



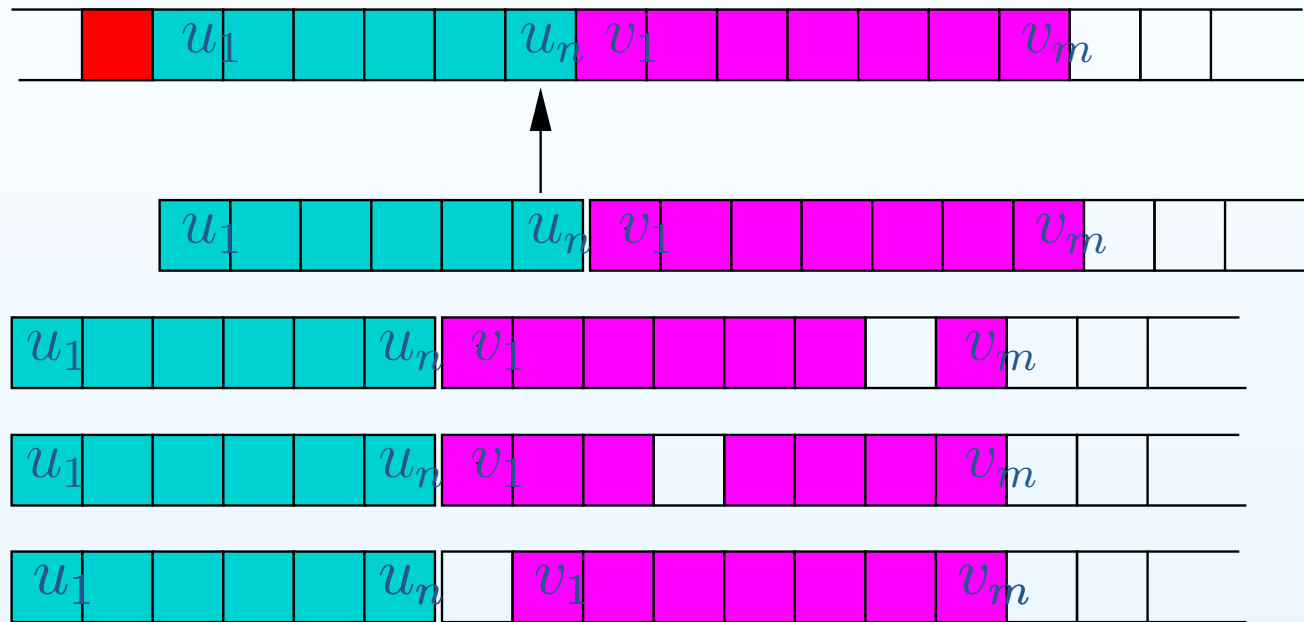
Equivalence entre Machine de Turing et Machine de Turing à semi-ruban



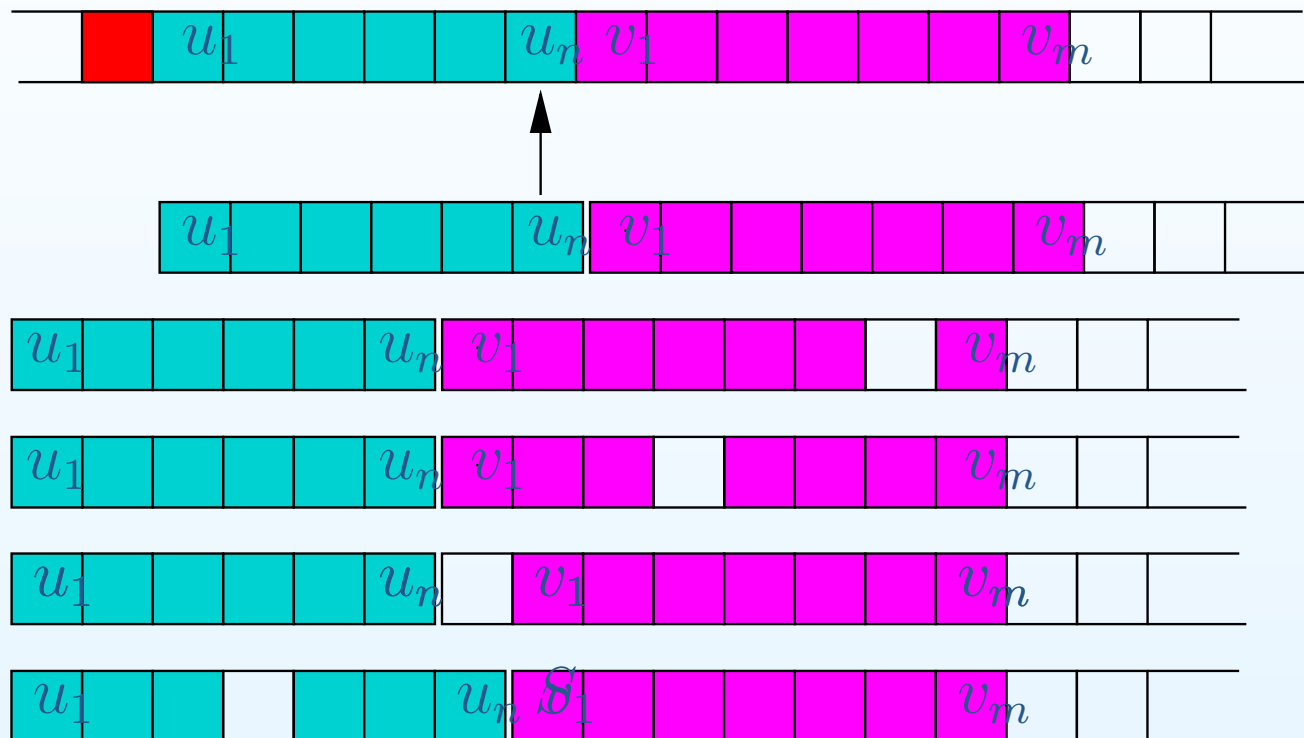
Equivalence entre Machine de Turing et Machine de Turing à semi-ruban



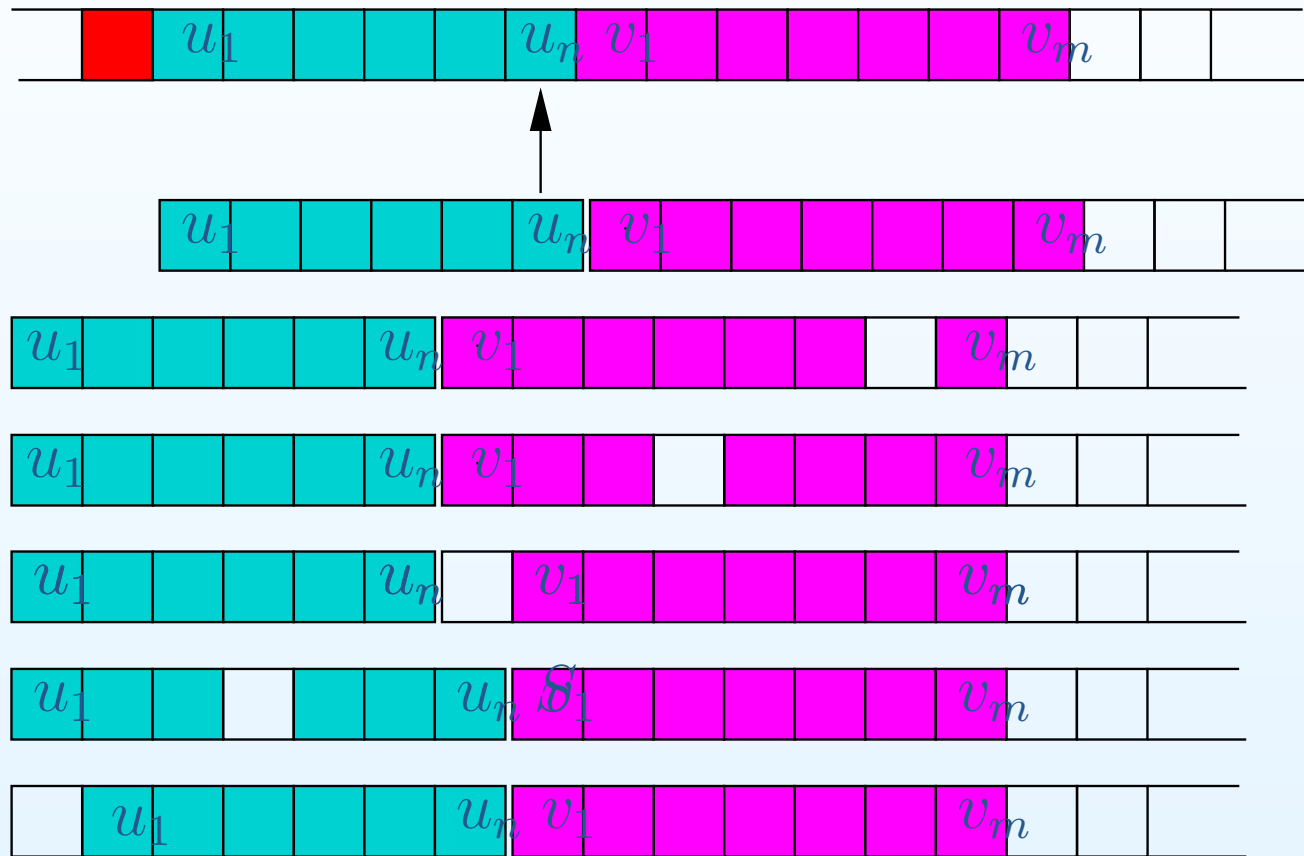
Equivalence entre Machine de Turing et Machine de Turing à semi-ruban



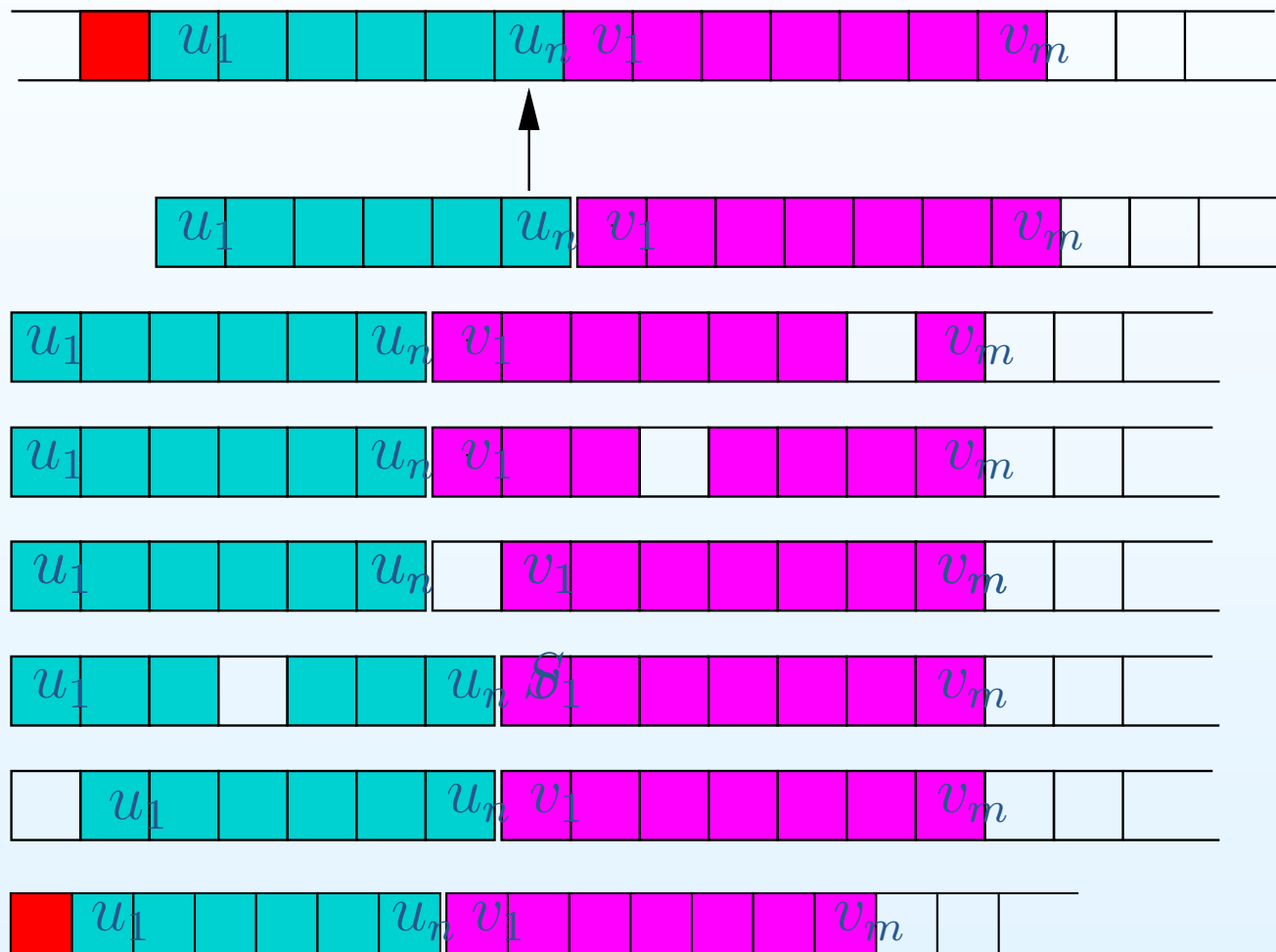
Equivalence entre Machine de Turing et Machine de Turing à semi-ruban



Equivalence entre Machine de Turing et Machine de Turing à semi-ruban



Equivalence entre Machine de Turing et Machine de Turing à semi-ruban



MT et Automates à 2 piles

Théorème Tout langage récursivement énumérable (décidable) est reconnaissable (décidable) par un automates à 2 piles et inversement.

Modèles équivalents

- La définition des machines de Turing peut être modifiée sans changer leur pouvoir de reconnaissance :
 1. il peut y avoir plusieurs rubans
 2. le(s) ruban(s) peuvent être semi-infinis
 3. la tête de lecture/écriture peut ou ne peut pas rester stationnaire
 4. on peut écrire ou non le symbole blanc
 5. la fonction de transition peut être déterministe ou pas
- à vouloir améliorer les MT, on retombe toujours sur des machines reconnaissant la même classe de langages: elles semblent englober toute idée de **procédure effective**

Machine à compteurs

- Un ruban en lecture seulement sur lequel se trouve le mot d'entrée.
- Des variables $x_1, \dots, x_n$ qui prennent des valeurs dans $\mathbb{N}$.
- Les instructions: d est une direction dans $\{G, D, I\}$.
 - $\ell : x := x + 1 \{ \text{lire } a, d \} \text{ goto } \ell'$
 - $\ell : x := x - 1 \{ \text{lire } a d \} \text{ goto } \ell'$
 - **if** $x = 0$ **then goto** ℓ' **else goto** ℓ'' .
 - Un label **fin**
 - Configuration initiale donnée par un label ℓ_0 , le pointeur sur mot d'entrée et les variables initialisées à 0.
 - On accepte le mot d'entrée si on atteint le label **fin**.

MT et Machine à compteurs

Théorème

1. Tout langage récursivement énumérable (décidable) est reconnaissable (décidable) par une machine à 2 compteurs, et inversement.
2. Un langage reconnu (décidé) par une machine à n compteurs est reconnu (décidé) une machine à 2 compteurs.

Preuve

1. Une machine à 2 piles avec $\Gamma = \{0, 1\}$ peut être simulée par une machine à 2 compteurs : empiler 0 = $2x$; empiler 1 = $2x+1$; dépiler 0 = $\frac{x}{2}$; dépiler 1 = $\frac{x-1}{2}$.
2. Gödelisation.



Thèse de Church-Turing

- Thèse de Church-Turing: Les fonctions **calculables par une procédure effective** le sont par une machine de Turing.
- adopter cette thèse, c'est choisir une modélisation du concept de procédure effective; il n'y a pas de démonstration, ce n'est pas un théorème
- la théorie de la calculabilité se fonde sur cette thèse; elle permet aussi de montrer l'existence de fonctions non calculables, par un simple argument de diagonalisation

La non-décidabilité

- Existence de problèmes indécidable.
- Technique de la réduction pour montrer l'indécidabilité.

Correspondance entre problème et langage

Un problème est représenté par l'ensemble de ces instance positives.

Definition

- La classe de décidabilité R est l'ensemble des langages rékursifs.
- La classe de décidabilité RE est l'ensemble des langages rékursivement énumérables.
- La classe de décidabilité $co-RE$ est l'ensemble des langages dont le complément est rékursivement énumérables.



Liens entre les classe

Lemme La classe R est incluse dans la classe RE ($R \subseteq RE$). $\square$

Lemme La classe R est incluse dans la classe $co-RE$ ($R \subseteq co-RE$). $\square$

Lemme Un langage et son complément sont dans RE ssi il est dans R ($RE \cap co-RE = R$). $\square$

Propriétés de fermeture

Les classes R , RE et $co-RE$ sont fermées par:

- Intersection.
- Union.
- Concaténation.
- Kleene star (L^*).
- Homomorphisme et inverse homomorphisme.

La classe R est en plus fermée par l'opération du complément.

Lemme Le complément d'un langage dans R est dans R . $\square$

Fonctions calculable

Soit $f : \sigma_1^* \times \cdots \times \Sigma_n^* \rightarrow \Sigma$ une fonction.

On définit

$$L_f = \{u_1 \# \cdots \# u_n \# u \mid f(u_1, \dots, u_n) = u\}$$

sur l'alphabet $\{\#\} \cup \bigcup_{i=1}^m \Sigma_i \cup \Sigma$ où $\# \notin \bigcup_{i=1}^m \Sigma_i \cup \Sigma$.

Definition f est *calculable*, si $L_f \in R$.

f est *semi-calculable*, si $L_f \in RE$.



Concaténation d'une fonction et d'une machine

Soit $f : \Sigma_1^* \times \cdots \times \Sigma_n^* \rightarrow \Sigma$ une fonction calculable.

On peut définir la concaténation, $f; M$, de f et d'une machine de Turing M sur Σ^* comme la machine qui:

1. prend en entrée $u_1 \# \cdots \# u_n$
2. utilise la machine de L_f pour trouver u tel que $f(u_1, \cdots, u_n) = u$.
3. Lance M sur u et répond la même chose que M si celle-ci s'arrête.

Si f est totale alors la machine $f; M$ ne s'arrête pas uniquement si M ne s'arrête pas.

Equipotence

- Soient A et B deux ensembles. A et B sont **equipotents** ssi il existe une bijection de A vers B .
- On note : $A \approx B$.
- L'ensemble A est appelé **dénombrable** ssi $\mathbb{N} \approx A$.

Par exemple, $\mathbb{N}^2$ est dénombrable. Mais $\mathcal{P}(\mathbb{N})$ ne l'est pas.

Diagonalisation

L'idée de la preuve de Cantor que $\mathcal{P}(\mathbb{N})$ n'est pas diagonalisable. Supposons au contraire que $\mathcal{P}(\mathbb{N})$ l'est. Soit $f : \mathbb{N} \rightarrow \mathcal{P}(\mathbb{N})$ une bijection.

	0	1	2	3	4	...
$f(0)$	0	1	1	0	0	...
$f(1)$	1	1	0	1	1	...
$f(2)$	0	0	1	0	0	...
$f(3)$	1	1	1	0	0	...
$f(4)$	1	1	0	0	0	...
$\vdots$						

Soit $D = \{i \in \mathbb{N} \mid i \in f(i)\}$, donc les i avec 1 sur la diagonale. Soit $\bar{D}$ le complément: $\bar{D} = \{i \in \mathbb{N} \mid i \notin f(i)\}$. Comme $D' \in \mathcal{P}(\mathbb{N})$, il existe i_0 avec $f(i_0) = D'$. Or, $i_0 \in f(i_0)$ ssi $i_0 \in D'$ ssi $i_0 \notin f(i_0)$.

Théorème de Cantor

Théorème:

- $\mathcal{P}(\mathbb{N})$ n'est pas dénombrable.
- $\mathcal{P}(A)$ et A ne sont pas equipotents.

Théorème de Cantor

Théorème:

- $\mathcal{P}(\mathbb{N})$ n'est pas dénombrable.
- $\mathcal{P}(A)$ et A ne sont pas equipotents.

Preuve

- Démontrons que A et $\mathcal{P}(A)$ ne sont pas équipotents.
- Il faut donc montrer qu'il n'y a pas de bijection entre A et $\mathcal{P}(A)$.

Théorème de Cantor

Théorème:

- $\mathcal{P}(\mathbb{N})$ n'est pas dénombrable.
- $\mathcal{P}(A)$ et A ne sont pas equipotents.

Preuve

- Soit $X = \{x \in A \mid x \notin f(x)\}$. Donc $X \subseteq A$ et $X \in \mathcal{P}(A)$.
- A cause de la surjection de f , il existe x_0 tel que $f(x_0) = X$
- Maintenant deux cas :



Théorème de Cantor

Théorème:

- $\mathcal{P}(\mathbb{N})$ n'est pas dénombrable.
- $\mathcal{P}(A)$ et A ne sont pas equipotents.

Preuve

- Soit $X = \{x \in A \mid x \notin f(x)\}$. Donc $X \subseteq A$ et $X \in \mathcal{P}(A)$.
- A cause de la surjection de f , il existe x_0 tel que $f(x_0) = X$
- Maintenant deux cas :
 1. $x_0 \in X$



Théorème de Cantor

Théorème:

- $\mathcal{P}(\mathbb{N})$ n'est pas dénombrable.
- $\mathcal{P}(A)$ et A ne sont pas equipotents.

Preuve

- Soit $X = \{x \in A \mid x \notin f(x)\}$. Donc $X \subseteq A$ et $X \in \mathcal{P}(A)$.
- A cause de la surjection de f , il existe x_0 tel que $f(x_0) = X$
- Maintenant deux cas :
 1. $x_0 \in X \implies x_0 \notin (f(x_0) = X)$ contradiction.



Théorème de Cantor

Théorème:

- $\mathcal{P}(\mathbb{N})$ n'est pas dénombrable.
- $\mathcal{P}(A)$ et A ne sont pas equipotents.

Preuve

- Soit $X = \{x \in A \mid x \notin f(x)\}$. Donc $X \subseteq A$ et $X \in \mathcal{P}(A)$.
- A cause de la surjection de f , il existe x_0 tel que $f(x_0) = X$
- Maintenant deux cas :
 1. $x_0 \in X \implies x_0 \notin (f(x_0) = X)$ contradiction.
 2. $x_0 \notin X$



Théorème de Cantor

Théorème:

- $\mathcal{P}(\mathbb{N})$ n'est pas dénombrable.
- $\mathcal{P}(A)$ et A ne sont pas equipotents.

Preuve

- Soit $X = \{x \in A \mid x \notin f(x)\}$. Donc $X \subseteq A$ et $X \in \mathcal{P}(A)$.
- A cause de la surjection de f , il existe x_0 tel que $f(x_0) = X$
- Maintenant deux cas :
 1. $x_0 \in X \implies x_0 \notin (f(x_0) = X)$ contradiction.
 2. $x_0 \notin X \implies x_0 \in f(x_0)$



Théorème de Cantor

Théorème:

- $\mathcal{P}(\mathbb{N})$ n'est pas dénombrable.
- $\mathcal{P}(A)$ et A ne sont pas equipotents.

Preuve

- Soit $X = \{x \in A \mid x \notin f(x)\}$. Donc $X \subseteq A$ et $X \in \mathcal{P}(A)$.
- A cause de la surjection de f , il existe x_0 tel que $f(x_0) = X$
- Maintenant deux cas :
 1. $x_0 \in X \implies x_0 \notin (f(x_0) = X)$ contradiction.
 2. $x_0 \notin X \implies x_0 \in f(x_0) \implies x_0 \in X$ contradiction.



L'ensemble des MT est dénombrable

Soient $\Sigma = \{0, 1\}$ et $\Gamma = \{0, 1, B\}$.

- L'ensemble des mots sur Σ est dénombrable.
- L'ensemble des machines de Turing avec alphabet d'entrée Σ et alphabet de ruban Γ est dénombrable.

Une machine de Turing peut-être codée comme un mot dans $\Sigma = \{0, 1\}$. Dans la suite M dénote ce codage. De la même manière un mot $u \in \Sigma^*$ peut être codé comme un mot dans $\Sigma = \{0, 1\}$.

Dans la suite on dénote par M_i la machine d'indice $i \in \mathbb{N}$ et par w_i le mot d'indice $i \in \mathbb{N}$.

Un problème indécidable

Soit $K = \{w_i \in \Sigma^* \mid w_i \in L(M_i)\}$.

Théorème L'ensemble (problème) K est indécidable, i.e., $K \notin R$.

Preuve On considère l'ensemble $\bar{K} = \{w_i \mid w_i \notin L(M_i)\}$.

On va montrer que $\bar{K}$ n'est pas dans RE.

Supposons le contraire. Alors, il existe une machine de Turing d'indice i_0 telle que $L(M_{i_0}) = \bar{K}$. Mais alors on a la contradiction suivante : $w_{i_0} \in L(M_{i_0})$ ssi $w_{i_0} \in \bar{K}$ ssi $w_{i_0} \notin L(M_{i_0})$.

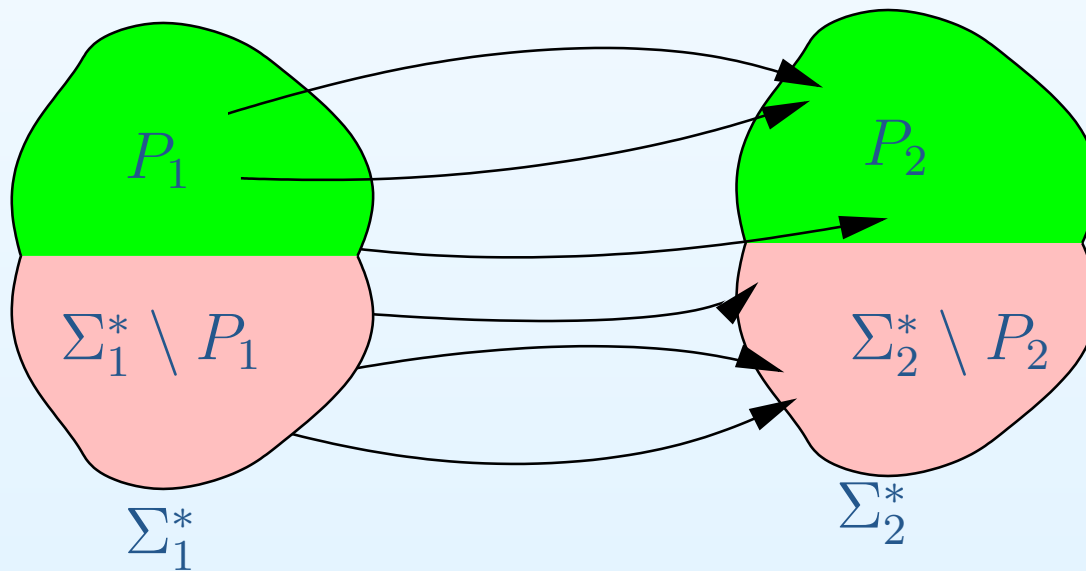
Donc $\bar{K}$ n'est pas dans RE. Par conséquent, K n'est pas dans R. □

Technique de la réduction

Soit P_1 et P_2 deux problèmes (langages). Soient Σ_1 et Σ_2 les alphabets de P_1 et P_2 .

Réduire P_1 à P_2 consiste à montrer l'existence d'une fonction *totale* et *calculable* de Σ_1^* vers Σ_2^* telle que

$$u \in P_1 \text{ ssi } f(u) \in P_2.$$



Reduction

Si on peut réduire P_1 à P_2 alors on peut conclure :

- Si P_1 n'est pas dans R alors P_2 n'est pas dans R. Pareil pour RE.
- Si P_2 est dans R alors P_1 est dans R.

En effet, supposons que la machine M décide P_2 . Alors, la machine $f; M$ décide P_1 . Pareil, si $P_2 \in \text{RE}$.

Le langage universel

Le langage universel est :

$$LU = \{M\#w \mid w \in L(M)\}$$

Théorème $LU \in RE \setminus R$.

Le langage universel

Preuve $LU \notin R$: soit f la fonction qui à w associe le mot $M_i \# w_i$ tel que i est l'indice de w , c.a.d. $w_i = w$. Alors f est totale et on a $w \in K$ ssi $f(w) \in LU$. En plus $L_f = \{w \# M_i \# w_i \mid f(w) = M_i \# w_i\} \in R$. En effet, la machine suivante décide L_f :

1. Entrée: u .
2. Si u n'est pas de la forme $u_1 \# u_2 \# u_3$, ne pas accepter.
3. Sinon, chercher $i \in \mathbb{N}$ tel que $u_1 = w_i$. Ceci peut être fait de la manière suivante : on commence avec $i = 0$ et on incrémente i de 1 tant que $w_i \neq u_1$. Le test $w_i \neq u_1$ est clairement décidable.
4. Déterminer M_i .
5. Vérifier $u_2 = M_i \wedge u_3 = u_1$.

$LU \in RE$: en utilisant la machine universelle.



Pblème de l'arrêt

Le problème

$$H = \{M\#w \mid M \text{ s'arrête sur } w\}$$

Théorème $H \in \text{RE} \setminus \text{R}$.

Preuve $H \notin \text{R}$:

$f(M\#w) = M'\#w$ où M' est la machine suivante :

1. Entrée: u .
2. Exécuter M sur w .
3. Si M s'arrête dans un état non-acceteur, alors entrer dans une boucle infinie.

On a : $M\#w \in LU$ ssi $f(M\#w) \in H$, f est totale et calculable.
 $H \in \text{RE}$: en utilisant la machine universelle et en allant dans un état acceteur quand l'exécution de M termine. □

Théorème de Rice

Soit $\Sigma = \{0, 1\}$.

Une propriété non-triviale sur les machines de Turing est un sous-ensemble $P \subseteq \Sigma^*$ tel que :

1. Pour toutes machines M et N :
 $L(M) = L(N) \Rightarrow [M \in P \Leftrightarrow N \in P]$.
2. Il existe $M_0, M_1 : M_0 \in P$ et $M_1 \notin P$.

Théorème de Rice

Théorème [Rice] Si P est une propriété non-triviale sur le machine de Turing alors $P \notin \mathcal{R}$:

$$P = \{M \mid M \in P\} \notin \mathcal{R}$$

Théorème de Rice : Preuve

La preuve est par réduction de H sur P . Supposons que P satisfait la propriété suivante: pour toute MT M , $L(M) = \emptyset \Rightarrow M \notin P$. Sinon on montre que $\Sigma^* \setminus P \notin R$. Soit M_0 une machine telle que $M_0 \in P$. Soit f la fonction telle que $f(M\#w)$ est une machine de Turing avec

$$L(f(M\#w)) = \begin{cases} L(M_0); & \text{si } M\#w \in H \\ \emptyset & ; \text{sinon} \end{cases}$$

Alors, $M\#w \in H$ ssi $f(M\#w) \in P$ et f est totale. La machine $f(M\#u)$ se comporte de la manière suivante:

1. Sauvegarder l'entrée u dans un deuxième ruban.
2. Exécuter M sur w .
3. Si M s'arrête, alors exécuter M_0 sur u et répondre la même chose.

Exemples

1. $\{M \mid M \text{ contient 5 états}\}.$
2. $\{M \mid M \text{ n'a aucun état accepteur }\}.$
3. $\{M \mid L(M) = 0^*1^*\}.$
4. $\{M \mid |L(M)| \equiv_2 0\}.$
5. $\{M \mid L(M) \text{ est infini}\}.$

Exemples

1. $\{M \mid L(M) \subseteq L_0\}$ pour un langage fixé L_0 .
2. $\{M \mid L(M) \supseteq L_0\}$ pour un langage fixé L_0 .
3. $\{M \mid L(M) = L_0\}$ pour un langage fixé $L_0 \in \text{RE}$.
4. $\{M \mid L(M) \neq L_0\}$ pour un langage fixé $L_0 \in \text{RE}$.
5. $\{M \mid L(M) = L_0\}$ pour un langage fixé $L_0 \in \text{co-RE}$.
6. $\{M \mid L(M) \neq L_0\}$ pour un langage fixé $L_0 \in \text{co-RE}$.
7. $\{M \mid L(M) = L_0\}$.
8. $\{M \mid \forall u \in L(M) \mid u \mid \leq k\}$, pour $k \in \mathbb{N}$ (fixé).
9. $\{M \# o^k \mid k \in \mathbb{N} \wedge \forall u \in L(M) \mid u \mid \leq k\}$.

Induction

Techniques de preuves

- Tiers exclu.
- Preuve par contradiction.
- Preuve par contraposition.
- Preuve par récurrence.
- Preuve par induction complète :

Techniques de preuves

- Tiers exclu.
- Preuve par contradiction.
- Preuve par contraposition.
- Preuve par récurrence.
- **Preuve par induction complète :**
Pour montrer $\forall n \in \mathbb{N} \cdot P(n)$ par induction complète, il faut montrer:

$$(\forall n \in \mathbb{N}) \cdot [(\forall m < n \cdot P(m)) \Rightarrow P(n)].$$

Exemple

Essayons de montrer la propriété $P(n)$ suivante :

$$\forall n \in \mathbb{N} \cdot n \geq 8 \Rightarrow \exists k \in \mathbb{N} \exists k' \in \mathbb{N} \cdot n = 3k + 5k'.$$

Exemple

Essayons de montrer la propriété $P(n)$ suivante :

$$\forall n \in \mathbb{N} \cdot n \geq 8 \Rightarrow \exists k \in \mathbb{N} \exists k' \in \mathbb{N} \cdot n = 3k + 5k'.$$

On distingue 5 cas :

Exemple

Essayons de montrer la propriété $P(n)$ suivante :

$$\forall n \in \mathbb{N} \cdot n \geq 8 \Rightarrow \exists k \in \mathbb{N} \exists k' \in \mathbb{N} \cdot n = 3k + 5k'.$$

On distingue 5 cas :

1. $n < 8$: trivial.

Exemple

Essayons de montrer la propriété $P(n)$ suivante :

$$\forall n \in \mathbb{N} \cdot n \geq 8 \Rightarrow \exists k \in \mathbb{N} \exists k' \in \mathbb{N} \cdot n = 3k + 5k'.$$

On distingue 5 cas :

1. $n < 8$: trivial.
2. $n = 8$: on pose $k = k' = 1$.

Exemple

Essayons de montrer la propriété $P(n)$ suivante :

$$\forall n \in \mathbb{N} \cdot n \geq 8 \Rightarrow \exists k \in \mathbb{N} \exists k' \in \mathbb{N} \cdot n = 3k + 5k'.$$

On distingue 5 cas :

1. $n < 8$: trivial.
2. $n = 8$: on pose $k = k' = 1$.
3. $n = 9$: on pose $k = 3$ et $k' = 0$.

Exemple

Essayons de montrer la propriété $P(n)$ suivante :

$$\forall n \in \mathbb{N} \cdot n \geq 8 \Rightarrow \exists k \in \mathbb{N} \exists k' \in \mathbb{N} \cdot n = 3k + 5k'.$$

On distingue 5 cas :

1. $n < 8$: trivial.
2. $n = 8$: on pose $k = k' = 1$.
3. $n = 9$: on pose $k = 3$ et $k' = 0$.
4. $n = 10$: on pose $k = 0$ et $k' = 2$.

Exemple

Essayons de montrer la propriété $P(n)$ suivante :

$$\forall n \in \mathbb{N} \cdot n \geq 8 \Rightarrow \exists k \in \mathbb{N} \exists k' \in \mathbb{N} \cdot n = 3k + 5k'.$$

On distingue 5 cas :

1. $n < 8$: trivial.
2. $n = 8$: on pose $k = k' = 1$.
3. $n = 9$: on pose $k = 3$ et $k' = 0$.
4. $n = 10$: on pose $k = 0$ et $k' = 2$.
5. $n \geq 11$: Soit $n \geq 11$ tel que $\forall m < n \cdot P(m)$. Alors, on a $P(n - 3)$. Comme $n - 3 \geq 8$, il existe $k \in \mathbb{N}$ et $k' \in \mathbb{N}$ tels que $n - 3 = 3k + 5k'$. Alors, $n = 3(k + 1) + 5k'$.

Fonctions Continues

Soit U un ensemble non-vidé que nous appelons *univers*

Soit $F : \mathcal{P}(U) \rightarrow \mathcal{P}(U)$ une fonction.

On dit que F est $\subseteq$ -monotone ssi

$$\forall A, B \in \mathcal{P}(U) \cdot A \subseteq B \Rightarrow F(A) \subseteq F(B).$$

On dit que F est continue, si les deux conditions suivantes sont satisfaites:

1. F est monotone
2. Pour toute chaîne $(X_i)_{i \in \mathbb{N}}$ avec $X_i \subseteq X_{i+1}$, pour tout $i \in \mathbb{N}$, on a

$$F\left(\bigcup_{i \in \mathbb{N}} X_i\right) = \bigcup_{i \in \mathbb{N}} F(X_i).$$

Exemples

Soit $U = \mathbb{N}$.

- L'identité sur $\mathcal{P}(\mathbb{N})$ est continue.
- Toute fonction constante est continue.
- La fonction qui à X lui associe $\mathbb{N} \setminus X$ n'est pas monotone.
- Soit $F : \mathcal{P}(\mathbb{N}) \rightarrow \mathcal{P}(\mathbb{N})$ définie par:

$$F(X) = \begin{cases} \emptyset & ; \text{ si } X \text{ est fini} \\ X & ; \text{ sinon} \end{cases}$$

Alors F n'est pas continue.

En effet, Soit $X_i = \{j \in \mathbb{N} \mid j \leq i\}$, pour $i \in \mathbb{N}$. Alors, $(X_i)_{i \in \mathbb{N}}$ est une chaîne et on a

$$\bigcup_{i \in \mathbb{N}} F(X_i) = \emptyset \text{ mais } F\left(\bigcup_{i \in \mathbb{N}} X_i\right) = F(\mathbb{N}) = \mathbb{N}.$$

Définition Equationnelle

Soit $F : \mathcal{P}(U) \rightarrow \mathcal{P}(U)$. Alors

$$X = F(X)$$

définit un ensemble E qui est la plus petite solution de cette équation.

Mais une telle équation a-t-elle toujours une solution? et une plus petite?

Théorème de Kleene

On définit $F^0(X) = X$ et $F^{i+1}(X) = F(F^i(X))$, pour tout $i \in \mathbb{N}$.

Théorème : (Kleene)

Soit $F : \mathcal{P}(U) \rightarrow \mathcal{P}(U)$ une fonction continue.

Alors

$$\bigcup_{i \in \mathbb{N}} F^i(\emptyset)$$

est la plus petite solution dans $\mathcal{P}(U)$ de l'équation $X = F(X)$.

En d'autres mots $\bigcup_{i \in \mathbb{N}} F^i(\emptyset)$ est le plus petit point-fixe de F .

Remarques: F peut avoir plusieurs point-fixes.

Preuve du Théorème de Kleene

On montre que $\bigcup_{i \in \mathbb{N}} F^i(\emptyset)$ est la plus petite solution de $X = F(X)$.

Montrons que c'est une solution:

$$F\left(\bigcup_{i \in \mathbb{N}} F^i(\emptyset)\right) = \bigcup_{i \in \mathbb{N}} F(F^i(\emptyset)) = \bigcup_{i \geq 1} F^i(\emptyset) = \emptyset \cup \bigcup_{i \geq 1} F^i(\emptyset) = \bigcup_{i \in \mathbb{N}} F^i(\emptyset).$$

Preuve du Théorème de Kleene

On montre que $\bigcup_{i \in \mathbb{N}} F^i(\emptyset)$ est la plus petite solution de $X = F(X)$.

Montrons $\bigcup_{i \in \mathbb{N}} F^i(\emptyset)$ est la plus petite solution.

Soit Y une solution. On montre par récurrence sur i :

$$F^i(\emptyset) \subseteq Y.$$

- Base de la récurrence: $i = 0$. Alors, $F^i(\emptyset) = \emptyset \subseteq Y$.
- Récurrence: Supposons $F^i(\emptyset) \subseteq Y$. On montre $F^{i+1}(\emptyset) \subseteq Y$.

Comme $F^i(\emptyset) \subseteq Y$, par monotonie de F , on a

$$F^{i+1}(\emptyset) = F(F^i(\emptyset)) \subseteq F(Y).$$

Donc, $F^{i+1}(\emptyset) \subseteq Y$.

Preuve par Fermeture

Pour montrer

$$\bigcup_{i \in \mathbb{N}} F^i(\emptyset) \subseteq P$$

il suffit de montrer que P est fermé par F c.a.d.

$$F(P) = P$$

Exercice: Montrer qu'il suffit de montrer:

$$F(P) \subseteq P$$

Définition inductive

(cf. INF121)

Pour définir l'ensemble $A \subseteq U$ **inductivement**, on donne :

1. une base : un ensemble $B \subseteq U$
2. des règles d'induction : une fonction $f : U \rightarrow \mathcal{P}(U)$.

Qu'avons nous défini?

A la définition inductive (B, f) on associe la fonction $F : \mathcal{P}(U) \rightarrow \mathcal{P}(U)$ telle que:

$$X \mapsto B \cup X \cup \bigcup_{x \in X} f(x)$$

On montre que F est continue. Donc elle a un plus petit point-fixe. C'est ce que définit $E = (B, f)$.

Exemple

On définit $Pair \subseteq \mathbb{N}$ inductivement de la manière suivante :

1. **Base** : $B = \{0\}$.
2. **Induction** : $f : \mathbb{N} \rightarrow \mathcal{P}(\mathbb{N})$ telle que $f(n) = \{2 * n, n + 2\}$.

On montre $Pair = \{2k \mid k \in \mathbb{N}\}$.

Pour montrer $Pair \subseteq \{2k \mid k \in \mathbb{N}\}$, il suffit de montrer que

$$F(\{2k \mid k \in \mathbb{N}\}) = \{2k \mid k \in \mathbb{N}\}.$$

C.a.d.

$$B \cup \{2 * 2k, 2k + 2 \mid k \in \mathbb{N}\} \subseteq \{2k \mid k \in \mathbb{N}\}.$$

Principe de Preuve par Induction

Soit $F : \mathcal{P}(U) \rightarrow \mathcal{P}(U)$ telle que:

$$X \mapsto B \cup X \cup \bigcup_{x \in X} f(x).$$

Alors, pour montrer $F(P) = P$, il suffit de montrer:

- $B \subseteq P$ et
- $\forall x \in P \cdot f(x) \subseteq P$

Preuve par induction

On montre

$$Pair \subseteq \{2k \mid k \in \mathbb{N}\}$$

Preuve par induction :

1. **Cas de base** : $0 = 2 * 0 \in \{2k \mid k \in \mathbb{N}\}$.
2. **Pas d'induction** : On montre que $2 * n \in \{2k \mid k \in \mathbb{N}\}$ et $n + 2 \in \{2k \mid k \in \mathbb{N}\}$ en supposant $n \in \{2k \mid k \in \mathbb{N}\}$.
C'est trivial pour $2 * n$. Pour $n + 2$, on sait qu'il existe $k \in \mathbb{N}$ tel que $n = 2k$. Donc on a : $n + 2 = 2k + 2 = 2(k + 1)$.
Comme $k + 1 \in \mathbb{N}$, on a $n + 2 \in \{2k \mid k \in \mathbb{N}\}$.

Retour à l'exemple Pair

Il nous reste à montrer

$$\{2k \mid k \in \mathbb{N}\} \subseteq \textit{Pair}$$

On montre : $\forall k \in \mathbb{N} \cdot 2k \in \textit{Pair}$ par recurrence sur k .

- Base : $k = 0$. Alors, $2 * 0 = 0 \in \textit{Pair}$.
- Pas de recurrence : Soit $k \in \mathbb{N}$ tel que $2k \in \textit{Pair}$. Il faut montrer $2(k + 1) \in \textit{Pair}$.

On a $2(k + 1) = 2k + 2$. Comme $2k \in \textit{Pair}$, on a, par les règles d'induction, $2k + 2 \in \textit{Pair}$.

Exemple : Factoriel

On définit l'ensemble $Fact \subseteq \mathbb{N}^2$ inductivement :

- Base : $(0, 1) \in Fact$.
- Règles d'induction : $f(n, m) = \{(n + 1, m(n + 1))\}$.

On montre que $Fact$ est le graphe de la fonction factoriel.

Démonstration au tableau.

Définition inductive: une généralisation simple

Nous avons vu qu'une définition inductive (ou on dit aussi schéma inductif) est donné par une paire (B, f) avec $B \subseteq U$ et $f : U \rightarrow \mathcal{P}(U)$.

Il est facile et utile de généraliser cette définition telle que

$$f : U^n \rightarrow \mathcal{P}(U) \text{ avec } n \in \mathbb{N}.$$

A la définition inductive (B, f) on associe la fonction $F : \mathcal{P}(U) \rightarrow \mathcal{P}(U)$ telle que:

$$X \mapsto B \cup X \cup \bigcup_{(x_1, \dots, x_n) \in X^n} f(x)$$

Principe de preuve par induction

Soit A un ensemble défini inductivement par (B, f) . Pour montrer

$$\forall a \in A \cdot P(a) \text{ où } P \text{ est une propriété}$$

il suffit de montrer :

1. Cas de base : pour tout $b \in B$, on a $P(b)$.
2. Pâs d'induction : pour toute règle d'induction on a

$$\forall a_1, \dots, a_n \cdot \forall b \in f(a_1, \dots, a_n) \cdot P(a_1) \wedge \dots \wedge P(a_n) \Rightarrow P(b)$$

Techniques de preuves

Pour résumer les techniques de preuve vues en cours sont :

- Tiers exclu.
- Preuve par contradiction.
- Preuve par contraposition.
- Preuve par récurrence.
- Preuve par induction complète.
- Preuve par induction.