

# INDÉCIDABILITÉ

Calculabilité et Complexité

Paul Brunet

1. Introduction
2. Le problème de l'arrêt
3. Réductions et autres problèmes indécidables
4. Théorème de Rice




### 1. Introduction

### 2. Le problème de l'arrêt

### 3. Réductions et autres problèmes indécidables

### 4. Théorème de Rice

 Il existe des problèmes impossibles à résoudre avec un ordinateur.

- ✎ Il existe des problèmes impossibles à résoudre avec un ordinateur.
- ✎ Cela correspond à l'existence de langages indécidables :

définition (Langage Indécidable)

Un langage  $L \in \Sigma^*$  est indécidable si il n'existe pas de machine à décider  $\mathcal{M}$  telle que  $\mathcal{L}(\mathcal{M}) = L$ .

- 👉 Il existe des problèmes impossibles à résoudre avec un ordinateur.
- 👉 Cela correspond à l'existence de langages indécidables :

définition (Langage Indécidable)

Un langage  $L \in \Sigma^*$  est indécidable si il n'existe pas de machine à décider  $\mathcal{M}$  telle que  $\mathcal{L}(\mathcal{M}) = L$ .

- 👉 Il est en général difficile de montrer directement qu'un langage est indécidable.

- ✎ Il existe des problèmes impossibles à résoudre avec un ordinateur.
- ✎ Cela correspond à l'existence de langages indécidables :

## définition (Langage Indécidable)

Un langage  $L \in \Sigma^*$  est indécidable si il n'existe pas de machine à décider  $\mathcal{M}$  telle que  $\mathcal{L}(\mathcal{M}) = L$ .

- ✎ Il est en général difficile de montrer directement qu'un langage est indécidable.
- ✎ En revanche, on peut faire des réductions :

- ✎ Il existe des problèmes impossibles à résoudre avec un ordinateur.
- ✎ Cela correspond à l'existence de langages indécidables :

## définition (Langage Indécidable)

Un langage  $L \in \Sigma^*$  est indécidable si il n'existe pas de machine à décider  $\mathcal{M}$  telle que  $\mathcal{L}(\mathcal{M}) = L$ .

- ✎ Il est en général difficile de montrer directement qu'un langage est indécidable.
- ✎ En revanche, on peut faire des réductions :
  - Je sais que  $L_1 \subseteq \Sigma_1^*$  est indécidable.



- ☞ Il existe des problèmes impossibles à résoudre avec un ordinateur.
- ☞ Cela correspond à l'existence de langages indécidables :

## définition (Langage Indécidable)

Un langage  $L \in \Sigma^*$  est indécidable si il n'existe pas de machine à décider  $\mathcal{M}$  telle que  $\mathcal{L}(\mathcal{M}) = L$ .

- ☞ Il est en général difficile de montrer directement qu'un langage est indécidable.
- ☞ En revanche, on peut faire des réductions :
  - Je sais que  $L_1 \subseteq \Sigma_1^*$  est indécidable.
  - J'ai une fonction calculable  $\varphi : \Sigma_1^* \rightarrow \Sigma_2^*$ .

- ☞ Il existe des problèmes impossibles à résoudre avec un ordinateur.
- ☞ Cela correspond à l'existence de langages indécidables :

## définition (Langage Indécidable)

Un langage  $L \in \Sigma^*$  est indécidable si il n'existe pas de machine à décider  $\mathcal{M}$  telle que  $\mathcal{L}(\mathcal{M}) = L$ .

- ☞ Il est en général difficile de montrer directement qu'un langage est indécidable.
- ☞ En revanche, on peut faire des réductions :
  - Je sais que  $L_1 \subseteq \Sigma_1^*$  est indécidable.
  - J'ai une fonction calculable  $\varphi : \Sigma_1^* \rightarrow \Sigma_2^*$ .
  - Je sais que  $w \in L_1 \Leftrightarrow \varphi(w) \in L_2$ .

- ☞ Il existe des problèmes impossibles à résoudre avec un ordinateur.
- ☞ Cela correspond à l'existence de langages indécidables :

## définition (Langage Indécidable)

Un langage  $L \in \Sigma^*$  est indécidable si il n'existe pas de machine à décider  $\mathcal{M}$  telle que  $\mathcal{L}(\mathcal{M}) = L$ .

- ☞ Il est en général difficile de montrer directement qu'un langage est indécidable.
- ☞ En revanche, on peut faire des réductions :
  - Je sais que  $L_1 \subseteq \Sigma_1^*$  est indécidable.
  - J'ai une fonction calculable  $\varphi : \Sigma_1^* \rightarrow \Sigma_2^*$ .
  - Je sais que  $w \in L_1 \Leftrightarrow \varphi(w) \in L_2$ .
  - Alors, si  $L_2$  était décidable,  $L_1$  le serait aussi.

- ☞ Il existe des problèmes impossibles à résoudre avec un ordinateur.
- ☞ Cela correspond à l'existence de langages indécidables :

## définition (Langage Indécidable)

Un langage  $L \in \Sigma^*$  est indécidable si il n'existe pas de machine à décider  $\mathcal{M}$  telle que  $\mathcal{L}(\mathcal{M}) = L$ .

- ☞ Il est en général difficile de montrer directement qu'un langage est indécidable.
- ☞ En revanche, on peut faire des réductions :
  - Je sais que  $L_1 \subseteq \Sigma_1^*$  est indécidable.
  - J'ai une fonction calculable  $\varphi : \Sigma_1^* \rightarrow \Sigma_2^*$ .
  - Je sais que  $w \in L_1 \Leftrightarrow \varphi(w) \in L_2$ .
  - Alors, si  $L_2$  était décidable,  $L_1$  le serait aussi.
  - Comme  $L_1$  ne l'est pas,  $L_2$  ne l'est pas non plus.

- ☞ Il existe des problèmes impossibles à résoudre avec un ordinateur.
- ☞ Cela correspond à l'existence de langages indécidables :

## définition (Langage Indécidable)

Un langage  $L \in \Sigma^*$  est indécidable si il n'existe pas de machine à décider  $\mathcal{M}$  telle que  $\mathcal{L}(\mathcal{M}) = L$ .

- ☞ Il est en général difficile de montrer directement qu'un langage est indécidable.
- ☞ En revanche, on peut faire des réductions :
  - Je sais que  $L_1 \subseteq \Sigma_1^*$  est indécidable.
  - J'ai une fonction calculable  $\varphi : \Sigma_1^* \rightarrow \Sigma_2^*$ .
  - Je sais que  $w \in L_1 \Leftrightarrow \varphi(w) \in L_2$ .
  - Alors, si  $L_2$  était décidable,  $L_1$  le serait aussi.
  - Comme  $L_1$  ne l'est pas,  $L_2$  ne l'est pas non plus.
- ☞ On va donc commencer par trouver un premier langage indécidable.

### 1. Introduction



### 2. Le problème de l'arrêt

### 3. Réductions et autres problèmes indécidables

### 4. Théorème de Rice

☞ La machine universelle que l'on a construit précédemment reconnaît le langage suivant :

$$\mathcal{L}(\mathcal{U}) = L_{\in} := \{[\mathcal{M}, w]_{code} \mid w \in \mathcal{L}(\mathcal{M})\}.$$

- ✎ La machine universelle que l'on a construit précédemment reconnaît le langage suivant :

$$\mathcal{L}(\mathcal{U}) = L_{\in} := \{[\mathcal{M}, w]_{code} \mid w \in \mathcal{L}(\mathcal{M})\}.$$

- ✎ En revanche elle ne décide pas ce langage : si  $\mathcal{M}$  diverge sur l'entrée  $w$ , alors  $\mathcal{U}$  ne s'arrête jamais.



- ☞ La machine universelle que l'on a construit précédemment reconnaît le langage suivant :

$$\mathcal{L}(\mathcal{U}) = L_{\in} := \{[\mathcal{M}, w]_{code} \mid w \in \mathcal{L}(\mathcal{M})\}.$$

- ☞ En revanche elle ne décide pas ce langage : si  $\mathcal{M}$  diverge sur l'entrée  $w$ , alors  $\mathcal{U}$  ne s'arrête jamais.
- ☞ Peut-on faire mieux ? Autrement dit, le langage  $L_{\in}$  est-il décidable ?

# Première preuve d'indécidabilité

$$\mathcal{L}(\mathcal{U}) = L_{\in} := \{[\mathcal{M}, w]_{code} \mid w \in \mathcal{L}(\mathcal{M})\}.$$

On suppose  $L_{\in}$  décidable.

☞ Soit  $\mathcal{H}$  une machine décidant  $L_{\in}$ .

$$\mathcal{L}(\mathcal{U}) = L_{\in} := \{[\mathcal{M}, w]_{code} \mid w \in \mathcal{L}(\mathcal{M})\}.$$

On suppose  $L_{\in}$  décidable.

☞ Soit  $\mathcal{H}$  une machine décidant  $L_{\in}$ .

☞ On construit la machine  $\mathcal{Q}$  :

entrée:  $[\mathcal{M}]_{code}$   
simuler  $\mathcal{H}$  sur  $[\mathcal{M}, [\mathcal{M}]_{code}]_{code}$   
si  $\mathcal{H}$  accepte : rejeter  
si  $\mathcal{H}$  rejette : accepter

$$\mathcal{L}(\mathcal{U}) = L_{\in} := \{[\mathcal{M}, w]_{code} \mid w \in \mathcal{L}(\mathcal{M})\}.$$

On suppose  $L_{\in}$  décidable.

☞ Soit  $\mathcal{H}$  une machine décidant  $L_{\in}$ .

☞ On construit la machine  $\mathcal{Q}$  :

entrée:  $[\mathcal{M}]_{code}$   
simuler  $\mathcal{H}$  sur  $[\mathcal{M}, [\mathcal{M}]_{code}]_{code}$   
si  $\mathcal{H}$  accepte : rejeter  
si  $\mathcal{H}$  rejette : accepter

☞ On peut comprendre la machine  $\mathcal{Q}$  comme suit :

$$[\mathcal{M}]_{code} \in \mathcal{L}(\mathcal{M}) \Rightarrow [\mathcal{M}, [\mathcal{M}]_{code}]_{code} \in \mathcal{L}(\mathcal{H}) \Rightarrow [\mathcal{M}]_{code} \notin \mathcal{L}(\mathcal{Q}).$$

$$[\mathcal{M}]_{code} \notin \mathcal{L}(\mathcal{M}) \Rightarrow [\mathcal{M}, [\mathcal{M}]_{code}]_{code} \notin \mathcal{L}(\mathcal{H}) \Rightarrow [\mathcal{M}]_{code} \in \mathcal{L}(\mathcal{Q}).$$

$$\mathcal{L}(\mathcal{U}) = L_{\in} := \{[\mathcal{M}, w]_{code} \mid w \in \mathcal{L}(\mathcal{M})\}.$$

On suppose  $L_{\in}$  décidable.

☞ Soit  $\mathcal{H}$  une machine décidant  $L_{\in}$ .

☞ On construit la machine  $\mathcal{Q}$  :

entrée:  $[\mathcal{M}]_{code}$   
simuler  $\mathcal{H}$  sur  $[\mathcal{M}, [\mathcal{M}]_{code}]_{code}$   
si  $\mathcal{H}$  accepte : rejeter  
si  $\mathcal{H}$  rejette : accepter

☞ On peut comprendre la machine  $\mathcal{Q}$  comme suit :

$$[\mathcal{M}]_{code} \in \mathcal{L}(\mathcal{M}) \Rightarrow [\mathcal{M}, [\mathcal{M}]_{code}]_{code} \in \mathcal{L}(\mathcal{H}) \Rightarrow [\mathcal{M}]_{code} \notin \mathcal{L}(\mathcal{Q}).$$

$$[\mathcal{M}]_{code} \notin \mathcal{L}(\mathcal{M}) \Rightarrow [\mathcal{M}, [\mathcal{M}]_{code}]_{code} \notin \mathcal{L}(\mathcal{H}) \Rightarrow [\mathcal{M}]_{code} \in \mathcal{L}(\mathcal{Q}).$$

☞ Que se passe-t'il si on exécute  $\mathcal{Q}$  sur son propre code ?

$$[\mathcal{Q}]_{code} \in \mathcal{L}(\mathcal{Q}) \Rightarrow [\mathcal{Q}]_{code} \notin \mathcal{L}(\mathcal{Q}).$$

$$[\mathcal{Q}]_{code} \notin \mathcal{L}(\mathcal{Q}) \Rightarrow [\mathcal{Q}]_{code} \in \mathcal{L}(\mathcal{Q}).$$

# Première preuve d'indécidabilité

$$\mathcal{L}(\mathcal{U}) = L_{\in} := \{[\mathcal{M}, w]_{code} \mid w \in \mathcal{L}(\mathcal{M})\}.$$

On suppose  $L_{\in}$  décidable.

☞ Soit  $\mathcal{H}$  une machine décidant  $L_{\in}$ .

☞ On construit la machine  $\mathcal{Q}$  :

entrée:  $[\mathcal{M}]_{code}$   
simuler  $\mathcal{H}$  sur  $[\mathcal{M}, [\mathcal{M}]_{code}]_{code}$   
si  $\mathcal{H}$  accepte : rejeter  
si  $\mathcal{H}$  rejette : accepter

☞ On peut comprendre la machine  $\mathcal{Q}$  comme suit :

$$[\mathcal{M}]_{code} \in \mathcal{L}(\mathcal{M}) \Rightarrow [\mathcal{M}, [\mathcal{M}]_{code}]_{code} \in \mathcal{L}(\mathcal{H}) \Rightarrow [\mathcal{M}]_{code} \notin \mathcal{L}(\mathcal{Q}).$$

$$[\mathcal{M}]_{code} \notin \mathcal{L}(\mathcal{M}) \Rightarrow [\mathcal{M}, [\mathcal{M}]_{code}]_{code} \notin \mathcal{L}(\mathcal{H}) \Rightarrow [\mathcal{M}]_{code} \in \mathcal{L}(\mathcal{Q}).$$

☞ Que se passe-t'il si on exécute  $\mathcal{Q}$  sur son propre code ?

$$[\mathcal{Q}]_{code} \in \mathcal{L}(\mathcal{Q}) \Rightarrow [\mathcal{Q}]_{code} \notin \mathcal{L}(\mathcal{Q}).$$

$$[\mathcal{Q}]_{code} \notin \mathcal{L}(\mathcal{Q}) \Rightarrow [\mathcal{Q}]_{code} \in \mathcal{L}(\mathcal{Q}).$$

☞ C'est une contradiction !

# Première preuve d'indécidabilité

$$\mathcal{L}(\mathcal{U}) = L_{\in} := \{[\mathcal{M}, w]_{code} \mid w \in \mathcal{L}(\mathcal{M})\}.$$

On suppose  $L_{\in}$  décidable.

☞ Soit  $\mathcal{H}$  une machine décidant  $L_{\in}$ .

☞ On construit la machine  $\mathcal{Q}$  :

entrée:  $[\mathcal{M}]_{code}$   
simuler  $\mathcal{H}$  sur  $[\mathcal{M}, [\mathcal{M}]_{code}]_{code}$   
si  $\mathcal{H}$  accepte : rejeter  
si  $\mathcal{H}$  rejette : accepter

☞ On peut comprendre la machine  $\mathcal{Q}$  comme suit :

$$[\mathcal{M}]_{code} \in \mathcal{L}(\mathcal{M}) \Rightarrow [\mathcal{M}, [\mathcal{M}]_{code}]_{code} \in \mathcal{L}(\mathcal{H}) \Rightarrow [\mathcal{M}]_{code} \notin \mathcal{L}(\mathcal{Q}).$$

$$[\mathcal{M}]_{code} \notin \mathcal{L}(\mathcal{M}) \Rightarrow [\mathcal{M}, [\mathcal{M}]_{code}]_{code} \notin \mathcal{L}(\mathcal{H}) \Rightarrow [\mathcal{M}]_{code} \in \mathcal{L}(\mathcal{Q}).$$

☞ Que se passe-t'il si on exécute  $\mathcal{Q}$  sur son propre code ?

$$[\mathcal{Q}]_{code} \in \mathcal{L}(\mathcal{Q}) \Rightarrow [\mathcal{Q}]_{code} \notin \mathcal{L}(\mathcal{Q}).$$

$$[\mathcal{Q}]_{code} \notin \mathcal{L}(\mathcal{Q}) \Rightarrow [\mathcal{Q}]_{code} \in \mathcal{L}(\mathcal{Q}).$$

☞ C'est une contradiction !

☞ Donc  $L_{\in}$  est indécidable.

1. Introduction

2. Le problème de l'arrêt



3. Réductions et autres problèmes indécidables

4. Théorème de Rice



Soient  $\mathcal{P} = \langle I, P \rangle$  et  $\mathcal{P}' = \langle I', P' \rangle$  deux problèmes, avec  $\varphi : I \rightarrow \Sigma^*$  et  $\varphi' : I' \rightarrow \Sigma'^*$  leurs codages.

## définition (Réduction)

Une réduction de  $\mathcal{P}$  à  $\mathcal{P}'$  est une fonction **calculable**  $\rho$  de  $\Sigma^*$  vers  $\Sigma'^*$  telle que :

$$\forall i \in I, \exists i' \in I' : \rho(\varphi(i)) = \varphi'(i')$$

$$\forall i \in I, i \in P \Leftrightarrow \exists i' \in P' : \rho(\varphi(i)) = \varphi'(i').$$

Si une telle réduction existe, on dit que  **$\mathcal{P}$  se réduit à  $\mathcal{P}'$** , et on écrit  $\mathcal{P} \rightarrow \mathcal{P}'$ .

## théorème

Si  $\mathcal{P} \rightarrow \mathcal{P}'$ , alors :

- 1) Si  $\mathcal{P}'$  est reconnaissable alors  $\mathcal{P}$  est reconnaissable.
- 2) Si  $\mathcal{P}'$  est décidable alors  $\mathcal{P}$  est décidable.

## théorème

Si  $\mathcal{P} \rightarrow \mathcal{P}'$ , alors :

- 1) Si  $\mathcal{P}'$  est reconnaissable alors  $\mathcal{P}$  est reconnaissable.
- 2) Si  $\mathcal{P}'$  est décidable alors  $\mathcal{P}$  est décidable.

## corollaire

Si  $\mathcal{P} \rightarrow \mathcal{P}'$ , alors :

- 1) Si  $\mathcal{P}$  n'est pas reconnaissable alors  $\mathcal{P}'$  ne l'est pas non plus.
- 2) Si  $\mathcal{P}$  est indécidable alors  $\mathcal{P}'$  est également indécidable.

Supposons que  $\mathcal{P} \rightarrow \mathcal{P}'$  avec  $\mathcal{P}'$  est reconnaissable, on veut montrer que  $\mathcal{P}$  est reconnaissable.

Supposons que  $\mathcal{P} \rightarrow \mathcal{P}'$  avec  $\mathcal{P}'$  est reconnaissable, on veut montrer que  $\mathcal{P}$  est reconnaissable.

Comme  $\mathcal{P} \rightarrow \mathcal{P}'$ , on a  $\rho$  telle que :

$$\forall i \in I, \exists i' \in I' : \rho(\varphi(i)) = \varphi'(i')$$

$$\forall i \in I, i \in P \Leftrightarrow \exists i' \in P' : \rho(\varphi(i)) = \varphi'(i').$$

Comme  $\rho$  est calculable on a  $\mathcal{M}_\rho$  telle que :

$$\forall w \in \Sigma^*, \forall u, v \in \Sigma'^*, (\rho(w) = u \cdot v) \Leftrightarrow (\exists q \in F : q_0 w \rightarrow_{\mathcal{M}_\rho}^* u q v).$$

Comme  $\mathcal{P}'$  est reconnaissable on a  $\mathcal{M}_{\mathcal{P}'}$  telle que :

$$\forall i \in I', i \in P' \Leftrightarrow (\exists u, v, \exists q \in F : q_0 \varphi(i) \rightarrow_{\mathcal{M}_{\mathcal{P}'}}^* u q v).$$

## Preuve (suite)

On construit une machine  $\mathcal{M}$  pour reconnaître  $\mathcal{P}$  qui procédera comme suit :

☞ Sur l'entrée  $w$ , exécuter  $\mathcal{M}_\rho$ , pour obtenir  $\rho(w)$  sur le ruban.

## Preuve (suite)

On construit une machine  $\mathcal{M}$  pour reconnaître  $\mathcal{P}$  qui procédera comme suit :

- ✎ Sur l'entrée  $w$ , exécuter  $\mathcal{M}_\rho$ , pour obtenir  $\rho(w)$  sur le ruban.
- ✎ Ensuite, exécuter  $\mathcal{M}_{\mathcal{P}'}$  sur cette nouvelle entrée.

## Preuve (suite)

On construit une machine  $\mathcal{M}$  pour reconnaître  $\mathcal{P}$  qui procédera comme suit :

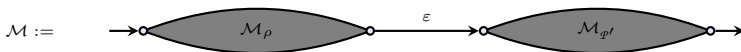
- ✎ Sur l'entrée  $w$ , exécuter  $\mathcal{M}_\rho$ , pour obtenir  $\rho(w)$  sur le ruban.
- ✎ Ensuite, exécuter  $\mathcal{M}_{\rho'}$  sur cette nouvelle entrée.
- ✎ Les états acceptants de  $\mathcal{M}$  sont ceux de  $\mathcal{M}_{\rho'}$ .



## Preuve (suite)

On construit une machine  $\mathcal{M}$  pour reconnaître  $\mathcal{P}$  qui procédera comme suit :

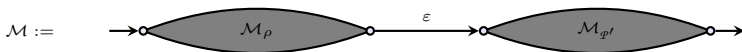
- ✎ Sur l'entrée  $w$ , exécuter  $\mathcal{M}_\rho$ , pour obtenir  $\rho(w)$  sur le ruban.
- ✎ Ensuite, exécuter  $\mathcal{M}_{p'}$  sur cette nouvelle entrée.
- ✎ Les états acceptants de  $\mathcal{M}$  sont ceux de  $\mathcal{M}_{p'}$ .



## Preuve (suite)

On construit une machine  $\mathcal{M}$  pour reconnaître  $\mathcal{P}$  qui procédera comme suit :

- ☞ Sur l'entrée  $w$ , exécuter  $\mathcal{M}_\rho$ , pour obtenir  $\rho(w)$  sur le ruban.
- ☞ Ensuite, exécuter  $\mathcal{M}_{\mathcal{P}^t}$  sur cette nouvelle entrée.
- ☞ Les états acceptants de  $\mathcal{M}$  sont ceux de  $\mathcal{M}_{\mathcal{P}^t}$ .

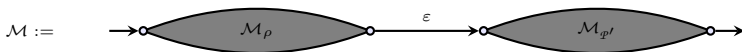


$$\mathcal{L}(\mathcal{M}) = \{w \in \Sigma^* \mid \rho(w) \in \mathcal{L}(\mathcal{M}_{\mathcal{P}^t})\}.$$

## Preuve (suite)

On construit une machine  $\mathcal{M}$  pour reconnaître  $\mathcal{P}$  qui procédera comme suit :

- ☞ Sur l'entrée  $w$ , exécuter  $\mathcal{M}_\rho$ , pour obtenir  $\rho(w)$  sur le ruban.
- ☞ Ensuite, exécuter  $\mathcal{M}_{\mathcal{P}'}'$  sur cette nouvelle entrée.
- ☞ Les états acceptants de  $\mathcal{M}$  sont ceux de  $\mathcal{M}_{\mathcal{P}'}'$ .

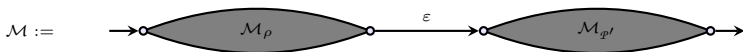


$$\mathcal{L}(\mathcal{M}) = \{w \in \Sigma^* \mid \rho(w) \in \mathcal{L}(\mathcal{M}_{\mathcal{P}'}')\}.$$

$$i \in \mathcal{P}$$

On construit une machine  $\mathcal{M}$  pour reconnaître  $\mathcal{P}$  qui procédera comme suit :

- ☞ Sur l'entrée  $w$ , exécuter  $\mathcal{M}_\rho$ , pour obtenir  $\rho(w)$  sur le ruban.
- ☞ Ensuite, exécuter  $\mathcal{M}_{\mathcal{P}'}$  sur cette nouvelle entrée.
- ☞ Les états acceptants de  $\mathcal{M}$  sont ceux de  $\mathcal{M}_{\mathcal{P}'}$ .

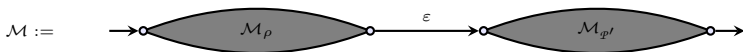


$$\mathcal{L}(\mathcal{M}) = \{w \in \Sigma^* \mid \rho(w) \in \mathcal{L}(\mathcal{M}_{\mathcal{P}'})\}.$$

$$i \in \mathcal{P} \Leftrightarrow \exists i' \in I' : i' \in \mathcal{P}' \text{ et } \rho(\varphi(i)) = \varphi'(i')$$

On construit une machine  $\mathcal{M}$  pour reconnaître  $\mathcal{P}$  qui procédera comme suit :

- ☞ Sur l'entrée  $w$ , exécuter  $\mathcal{M}_\rho$ , pour obtenir  $\rho(w)$  sur le ruban.
- ☞ Ensuite, exécuter  $\mathcal{M}_{\mathcal{P}'}$  sur cette nouvelle entrée.
- ☞ Les états acceptants de  $\mathcal{M}$  sont ceux de  $\mathcal{M}_{\mathcal{P}'}$ .

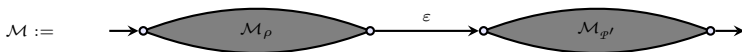


$$\mathcal{L}(\mathcal{M}) = \{w \in \Sigma^* \mid \rho(w) \in \mathcal{L}(\mathcal{M}_{\mathcal{P}'})\}.$$

$$\begin{aligned} i \in \mathcal{P} &\Leftrightarrow \exists i' \in I' : i' \in \mathcal{P}' \text{ et } \rho(\varphi(i)) = \varphi'(i') \\ &\Leftrightarrow \exists i' \in I' : \varphi'(i') \in \mathcal{L}(\mathcal{M}_{\mathcal{P}'}) \text{ et } \rho(\varphi(i)) = \varphi'(i') \end{aligned}$$

On construit une machine  $\mathcal{M}$  pour reconnaître  $\mathcal{P}$  qui procédera comme suit :

- ☞ Sur l'entrée  $w$ , exécuter  $\mathcal{M}_\rho$ , pour obtenir  $\rho(w)$  sur le ruban.
- ☞ Ensuite, exécuter  $\mathcal{M}_{\mathcal{P}'}$  sur cette nouvelle entrée.
- ☞ Les états acceptants de  $\mathcal{M}$  sont ceux de  $\mathcal{M}_{\mathcal{P}'}$ .

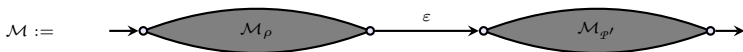


$$\mathcal{L}(\mathcal{M}) = \{w \in \Sigma^* \mid \rho(w) \in \mathcal{L}(\mathcal{M}_{\mathcal{P}'})\}.$$

$$\begin{aligned}i \in \mathcal{P} &\Leftrightarrow \exists i' \in I' : i' \in \mathcal{P}' \text{ et } \rho(\varphi(i)) = \varphi'(i') \\ &\Leftrightarrow \exists i' \in I' : \varphi'(i') \in \mathcal{L}(\mathcal{M}_{\mathcal{P}'}) \text{ et } \rho(\varphi(i)) = \varphi'(i') \\ &\Leftrightarrow \exists i' \in I' : \rho(\varphi(i)) \in \mathcal{L}(\mathcal{M}_{\mathcal{P}'}) \text{ et } \rho(\varphi(i)) = \varphi'(i')\end{aligned}$$

On construit une machine  $\mathcal{M}$  pour reconnaître  $\mathcal{P}$  qui procédera comme suit :

- ☞ Sur l'entrée  $w$ , exécuter  $\mathcal{M}_\rho$ , pour obtenir  $\rho(w)$  sur le ruban.
- ☞ Ensuite, exécuter  $\mathcal{M}_{\rho'}$  sur cette nouvelle entrée.
- ☞ Les états acceptants de  $\mathcal{M}$  sont ceux de  $\mathcal{M}_{\rho'}$ .

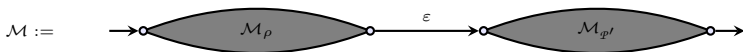


$$\mathcal{L}(\mathcal{M}) = \{w \in \Sigma^* \mid \rho(w) \in \mathcal{L}(\mathcal{M}_{\rho'})\}.$$

$$\begin{aligned}i \in \mathcal{P} &\Leftrightarrow \exists i' \in I' : i' \in P' \text{ et } \rho(\varphi(i)) = \varphi'(i') \\ &\Leftrightarrow \exists i' \in I' : \varphi'(i') \in \mathcal{L}(\mathcal{M}_{\rho'}) \text{ et } \rho(\varphi(i)) = \varphi'(i') \\ &\Leftrightarrow \exists i' \in I' : \rho(\varphi(i)) \in \mathcal{L}(\mathcal{M}_{\rho'}) \text{ et } \rho(\varphi(i)) = \varphi'(i') \\ &\Leftrightarrow \rho(\varphi(i)) \in \mathcal{L}(\mathcal{M}_{\rho'}) \text{ et } \exists i' \in I' : \rho(\varphi(i)) = \varphi'(i')\end{aligned}$$

On construit une machine  $\mathcal{M}$  pour reconnaître  $\mathcal{P}$  qui procédera comme suit :

- ☞ Sur l'entrée  $w$ , exécuter  $\mathcal{M}_\rho$ , pour obtenir  $\rho(w)$  sur le ruban.
- ☞ Ensuite, exécuter  $\mathcal{M}_{\mathcal{P}'}$  sur cette nouvelle entrée.
- ☞ Les états acceptants de  $\mathcal{M}$  sont ceux de  $\mathcal{M}_{\mathcal{P}'}$ .



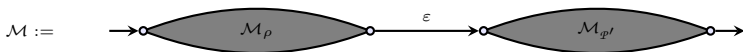
$$\mathcal{L}(\mathcal{M}) = \{w \in \Sigma^* \mid \rho(w) \in \mathcal{L}(\mathcal{M}_{\mathcal{P}'})\}.$$

$$\begin{aligned} i \in \mathcal{P} &\Leftrightarrow \exists i' \in I' : i' \in \mathcal{P}' \text{ et } \rho(\varphi(i)) = \varphi'(i') \\ &\Leftrightarrow \exists i' \in I' : \varphi'(i') \in \mathcal{L}(\mathcal{M}_{\mathcal{P}'}) \text{ et } \rho(\varphi(i)) = \varphi'(i') \\ &\Leftrightarrow \exists i' \in I' : \rho(\varphi(i)) \in \mathcal{L}(\mathcal{M}_{\mathcal{P}'}) \text{ et } \rho(\varphi(i)) = \varphi'(i') \\ &\Leftrightarrow \rho(\varphi(i)) \in \mathcal{L}(\mathcal{M}_{\mathcal{P}'}) \text{ et } \exists i' \in I' : \rho(\varphi(i)) = \varphi'(i') \\ &\Leftrightarrow \rho(\varphi(i)) \in \mathcal{L}(\mathcal{M}_{\mathcal{P}'}) \end{aligned}$$



On construit une machine  $\mathcal{M}$  pour reconnaître  $\mathcal{P}$  qui procédera comme suit :

- ☞ Sur l'entrée  $w$ , exécuter  $\mathcal{M}_\rho$ , pour obtenir  $\rho(w)$  sur le ruban.
- ☞ Ensuite, exécuter  $\mathcal{M}_{\mathcal{P}'}$  sur cette nouvelle entrée.
- ☞ Les états acceptants de  $\mathcal{M}$  sont ceux de  $\mathcal{M}_{\mathcal{P}'}$ .

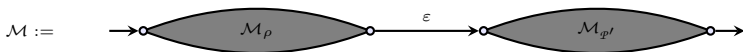


$$\mathcal{L}(\mathcal{M}) = \{w \in \Sigma^* \mid \rho(w) \in \mathcal{L}(\mathcal{M}_{\mathcal{P}'})\}.$$

$$\begin{aligned} i \in \mathcal{P} &\Leftrightarrow \exists i' \in I' : i' \in P' \text{ et } \rho(\varphi(i)) = \varphi'(i') \\ &\Leftrightarrow \exists i' \in I' : \varphi'(i') \in \mathcal{L}(\mathcal{M}_{\mathcal{P}'}) \text{ et } \rho(\varphi(i)) = \varphi'(i') \\ &\Leftrightarrow \exists i' \in I' : \rho(\varphi(i)) \in \mathcal{L}(\mathcal{M}_{\mathcal{P}'}) \text{ et } \rho(\varphi(i)) = \varphi'(i') \\ &\Leftrightarrow \rho(\varphi(i)) \in \mathcal{L}(\mathcal{M}_{\mathcal{P}'}) \text{ et } \exists i' \in I' : \rho(\varphi(i)) = \varphi'(i') \\ &\Leftrightarrow \rho(\varphi(i)) \in \mathcal{L}(\mathcal{M}_{\mathcal{P}'}) \\ &\Leftrightarrow \varphi(i) \in \mathcal{L}(\mathcal{M}). \end{aligned}$$

On construit une machine  $\mathcal{M}$  pour reconnaître  $\mathcal{P}$  qui procédera comme suit :

- 👉 Sur l'entrée  $w$ , exécuter  $\mathcal{M}_\rho$ , pour obtenir  $\rho(w)$  sur le ruban.
- 👉 Ensuite, exécuter  $\mathcal{M}_{\mathcal{P}'}$  sur cette nouvelle entrée.
- 👉 Les états acceptants de  $\mathcal{M}$  sont ceux de  $\mathcal{M}_{\mathcal{P}'}$ .



$$\mathcal{L}(\mathcal{M}) = \{w \in \Sigma^* \mid \rho(w) \in \mathcal{L}(\mathcal{M}_{\mathcal{P}'})\}.$$

$$\begin{aligned} i \in \mathcal{P} &\Leftrightarrow \exists i' \in I' : i' \in \mathcal{P}' \text{ et } \rho(\varphi(i)) = \varphi'(i') \\ &\Leftrightarrow \exists i' \in I' : \varphi'(i') \in \mathcal{L}(\mathcal{M}_{\mathcal{P}'}) \text{ et } \rho(\varphi(i)) = \varphi'(i') \\ &\Leftrightarrow \exists i' \in I' : \rho(\varphi(i)) \in \mathcal{L}(\mathcal{M}_{\mathcal{P}'}) \text{ et } \rho(\varphi(i)) = \varphi'(i') \\ &\Leftrightarrow \rho(\varphi(i)) \in \mathcal{L}(\mathcal{M}_{\mathcal{P}'}) \text{ et } \exists i' \in I' : \rho(\varphi(i)) = \varphi'(i') \\ &\Leftrightarrow \rho(\varphi(i)) \in \mathcal{L}(\mathcal{M}_{\mathcal{P}'}) \\ &\Leftrightarrow \varphi(i) \in \mathcal{L}(\mathcal{M}). \end{aligned}$$

**Donc  $\mathcal{P}$  est reconnaissable.**



# Problèmes Indécidables

## Exercice

Montrer que les problèmes suivants sont indécidables :

- 1) Arrêt sur le mot vide :  $\{\mathcal{M} \mid \varepsilon \in \mathcal{L}(\mathcal{M})\}$ .
- 2) Vacuité du langage :  $\{\mathcal{M} \mid \mathcal{L}(\mathcal{M}) = \emptyset\}$ .
- 3) Universalité :  $\{\mathcal{M} \mid \mathcal{L}(\mathcal{M}) = \Sigma^*\}$ .
- 4) Équivalence :  $\{\langle \mathcal{M}, \mathcal{M}' \rangle \mid \mathcal{L}(\mathcal{M}) = \mathcal{L}(\mathcal{M}')\}$ .

1. Introduction
2. Le problème de l'arrêt
3. Réductions et autres problèmes indécidables



4. Théorème de Rice

théorème (Henry Gordon Rice, 1953)

**Toute propriété non-triviale des langages des machines de Turing est indécidable.**

Une propriété est non-triviale si il existe au moins une instance positive et une instance négative.

théorème (Henry Gordon Rice, 1953)

Toute propriété **non-triviale** des langages des machines de Turing est indécidable.

Une propriété est non-triviale si il existe au moins une instance positive et une instance négative.

théorème (Henry Gordon Rice, 1953)

Toute propriété **non-triviale** des **langages des machines de Turing** est indécidable.

Une propriété des langages de machine de Turing est un ensemble  $P$  de machines de Turing telle que si deux machines reconnaissent le même langage, soit toutes les deux sont dans  $P$ , soit aucune des deux n'appartient à  $P$ .

## définition

Une propriété est non-triviale si il existe au moins une instance positive et une instance négative.

Autrement dit, soit  $\mathcal{P} = \langle I, P \rangle$  un problème.

☞ Soit  $P = I$ , c'est à dire que toutes les instances sont positives :



## définition

Une propriété est non-triviale si il existe au moins une instance positive et une instance négative.

Autrement dit, soit  $\mathcal{P} = \langle I, P \rangle$  un problème.

- ☞ Soit  $P = I$ , c'est à dire que toutes les instances sont positives :
- donc il n'existe pas d'instance négative,

## définition

Une propriété est non-triviale si il existe au moins une instance positive et une instance négative.

Autrement dit, soit  $\mathcal{P} = \langle I, P \rangle$  un problème.

- ☞ Soit  $P = I$ , c'est à dire que toutes les instances sont positives :
- donc il n'existe pas d'instance négative,
  - donc  $\mathcal{P}$  est triviale.

## définition

Une propriété est non-triviale si il existe au moins une instance positive et une instance négative.

Autrement dit, soit  $\mathcal{P} = \langle I, P \rangle$  un problème.

- ☞ Soit  $P = I$ , c'est à dire que toutes les instances sont positives :
  - donc il n'existe pas d'instance négative,
  - donc  $\mathcal{P}$  est triviale.
- ☞ Soit  $P = \emptyset$ , c'est à dire que toutes les instances sont négatives :

## définition

Une propriété est non-triviale si il existe au moins une instance positive et une instance négative.

Autrement dit, soit  $\mathcal{P} = \langle I, P \rangle$  un problème.

- ☞ Soit  $P = I$ , c'est à dire que toutes les instances sont positives :
  - donc il n'existe pas d'instance négative,
  - donc  $\mathcal{P}$  **est triviale**.
- ☞ Soit  $P = \emptyset$ , c'est à dire que toutes les instances sont négatives :
  - donc il n'existe pas d'instance positive,

## définition

Une propriété est non-triviale si il existe au moins une instance positive et une instance négative.

Autrement dit, soit  $\mathcal{P} = \langle I, P \rangle$  un problème.

- ☞ Soit  $P = I$ , c'est à dire que toutes les instances sont positives :
  - donc il n'existe pas d'instance négative,
  - donc  $\mathcal{P}$  est triviale.
- ☞ Soit  $P = \emptyset$ , c'est à dire que toutes les instances sont négatives :
  - donc il n'existe pas d'instance positive,
  - donc  $\mathcal{P}$  est triviale.

## définition

Une propriété est non-triviale si il existe au moins une instance positive et une instance négative.

Autrement dit, soit  $\mathcal{P} = \langle I, P \rangle$  un problème.

- ☞ Soit  $P = I$ , c'est à dire que toutes les instances sont positives :
  - donc il n'existe pas d'instance négative,
  - donc  $\mathcal{P}$  est triviale.
- ☞ Soit  $P = \emptyset$ , c'est à dire que toutes les instances sont négatives :
  - donc il n'existe pas d'instance positive,
  - donc  $\mathcal{P}$  est triviale.
- ☞ Soit  $P \notin \{I, \emptyset\}$  et donc :

## définition

Une propriété est non-triviale si il existe au moins une instance positive et une instance négative.

Autrement dit, soit  $\mathcal{P} = \langle I, P \rangle$  un problème.

- ☞ Soit  $P = I$ , c'est à dire que toutes les instances sont positives :
  - donc il n'existe pas d'instance négative,
  - donc  $\mathcal{P}$  est triviale.
- ☞ Soit  $P = \emptyset$ , c'est à dire que toutes les instances sont négatives :
  - donc il n'existe pas d'instance positive,
  - donc  $\mathcal{P}$  est triviale.
- ☞ Soit  $P \notin \{I, \emptyset\}$  et donc :
  - $I \setminus P \neq \emptyset$  (sinon, comme  $P \subseteq I$ , on aurait  $P = I$ ), et donc  $\exists i_- \in I \setminus P$ , autrement dit il existe une instance négative  $i_-$  ;

## définition

Une propriété est non-triviale si il existe au moins une instance positive et une instance négative.

Autrement dit, soit  $\mathcal{P} = \langle I, P \rangle$  un problème.

- ☞ Soit  $P = I$ , c'est à dire que toutes les instances sont positives :
  - donc il n'existe pas d'instance négative,
  - donc  $\mathcal{P}$  est triviale.
- ☞ Soit  $P = \emptyset$ , c'est à dire que toutes les instances sont négatives :
  - donc il n'existe pas d'instance positive,
  - donc  $\mathcal{P}$  est triviale.
- ☞ Soit  $P \notin \{I, \emptyset\}$  et donc :
  - $I \setminus P \neq \emptyset$  (sinon, comme  $P \subseteq I$ , on aurait  $P = I$ ), et donc  $\exists i_- \in I \setminus P$ , autrement dit il existe une instance négative  $i_-$  ;
  - $P \neq \emptyset$ , donc  $\exists i_+ \in P$ , autrement dit il existe une instance positive  $i_+$  ;



## définition

Une propriété est non-triviale si il existe au moins une instance positive et une instance négative.

Autrement dit, soit  $\mathcal{P} = \langle I, P \rangle$  un problème.

- ☞ Soit  $P = I$ , c'est à dire que toutes les instances sont positives :
  - donc il n'existe pas d'instance négative,
  - donc  $\mathcal{P}$  est triviale.
- ☞ Soit  $P = \emptyset$ , c'est à dire que toutes les instances sont négatives :
  - donc il n'existe pas d'instance positive,
  - donc  $\mathcal{P}$  est triviale.
- ☞ Soit  $P \notin \{I, \emptyset\}$  et donc :
  - $I \setminus P \neq \emptyset$  (sinon, comme  $P \subseteq I$ , on aurait  $P = I$ ), et donc  $\exists i_- \in I \setminus P$ , autrement dit il existe une instance négative  $i_-$  ;
  - $P \neq \emptyset$ , donc  $\exists i_+ \in P$ , autrement dit il existe une instance positive  $i_+$  ;
  - donc  $\mathcal{P}$  est non-triviale.

## définition

Une propriété est non-triviale si il existe au moins une instance positive et une instance négative.

Autrement dit, soit  $\mathcal{P} = \langle I, P \rangle$  un problème.

- ☞ Soit  $P = I$ , c'est à dire que toutes les instances sont positives :
  - donc il n'existe pas d'instance négative,
  - donc  $\mathcal{P}$  **est triviale**.
- ☞ Soit  $P = \emptyset$ , c'est à dire que toutes les instances sont négatives :
  - donc il n'existe pas d'instance positive,
  - donc  $\mathcal{P}$  **est triviale**.
- ☞ Soit  $P \notin \{I, \emptyset\}$  et donc :
  - $I \setminus P \neq \emptyset$  (sinon, comme  $P \subseteq I$ , on aurait  $P = I$ ), et donc  $\exists i_- \in I \setminus P$ , autrement dit il existe une instance négative  $i_-$  ;
  - $P \neq \emptyset$ , donc  $\exists i_+ \in P$ , autrement dit il existe une instance positive  $i_+$  ;
  - donc  $\mathcal{P}$  **est non-triviale**.
  
- ☞ Intuitivement, une propriété triviale est une question dont la réponse ne dépend pas de l'objet sur lequel elle porte.

## définition

Une propriété est non-triviale si il existe au moins une instance positive et une instance négative.

Autrement dit, soit  $\mathcal{P} = \langle I, P \rangle$  un problème.

- ☞ Soit  $P = I$ , c'est à dire que toutes les instances sont positives :
  - donc il n'existe pas d'instance négative,
  - donc  $\mathcal{P}$  **est triviale**.
- ☞ Soit  $P = \emptyset$ , c'est à dire que toutes les instances sont négatives :
  - donc il n'existe pas d'instance positive,
  - donc  $\mathcal{P}$  **est triviale**.
- ☞ Soit  $P \notin \{I, \emptyset\}$  et donc :
  - $I \setminus P \neq \emptyset$  (sinon, comme  $P \subseteq I$ , on aurait  $P = I$ ), et donc  $\exists i_- \in I \setminus P$ , autrement dit il existe une instance négative  $i_-$  ;
  - $P \neq \emptyset$ , donc  $\exists i_+ \in P$ , autrement dit il existe une instance positive  $i_+$  ;
  - donc  $\mathcal{P}$  **est non-triviale**.
  
- ☞ Intuitivement, une propriété triviale est une question dont la réponse ne dépend pas de l'objet sur lequel elle porte.
- ☞ Une telle propriété est donc **décidable** : si on nous donne une instance, on peut donner la réponse immédiatement, sans même examiner l'instance donnée en entrée !

Les problèmes suivants sont triviaux.

☞  $\langle \mathbb{N}, \{n \in \mathbb{N} \mid 7 \text{ est pair}\} \rangle$ .

☞  $\langle \mathbb{N}, \{n \in \mathbb{N} \mid 7 \text{ est impair}\} \rangle$ .

☞  $\langle \{\mathcal{M} \mid \mathcal{M} \text{ machine de Turing sur l'alphabet } \Sigma\}, \{\mathcal{M} \mid \mathcal{L}(\mathcal{M}) \subseteq \Sigma^*\} \rangle$ .

☞  $\langle \Sigma^*, \{w \in \Sigma^* \mid 0 \leq |w|\} \rangle$ .

## définition

Une propriété des langages de machine de Turing est un ensemble  $P$  de machines de Turing telle que si deux machines reconnaissent le même langage, soit toutes les deux sont dans  $P$ , soit aucune des deux n'appartient à  $P$ .

Formellement, un problème  $\mathcal{P} = \langle I, P \rangle$  est dit « de langage des machines de Turing » :

- 1) si  $I = \{\mathcal{M} \mid \mathcal{M} \text{ machines de Turing sur l'alphabet } \Sigma\}$ , pour un alphabet  $\Sigma$ ,
- 2) et si  $P$  vérifie la propriété suivante :

$$\forall \mathcal{M}_1, \mathcal{M}_2 \in I, \mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2) \Rightarrow (\mathcal{M}_1 \in P \Leftrightarrow \mathcal{M}_2 \in P).$$

Cette propriété est équivalente aux énoncés suivants :

- ☞  $\forall \mathcal{M}_1, \mathcal{M}_2 \in I, \mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2) \text{ et } \mathcal{M}_1 \in P \Rightarrow \mathcal{M}_2 \in P.$
- ☞  $\forall \mathcal{M}_1, \mathcal{M}_2 \in I, \mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2) \text{ et } \mathcal{M}_1 \notin P \Rightarrow \mathcal{M}_2 \notin P.$
- ☞  $\forall \mathcal{M}_1, \mathcal{M}_2 \in I, \mathcal{M}_1 \in P \text{ et } \mathcal{M}_2 \notin P \Rightarrow \mathcal{L}(\mathcal{M}_1) \neq \mathcal{L}(\mathcal{M}_2).$

## exercice

Montrer que les quatre propriétés ci-dessus sont équivalentes.

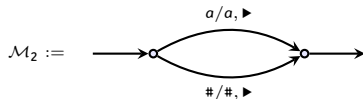
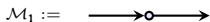
# Exemples

## Machines avec un nombre pair d'états

$$\forall \mathcal{M}_1, \mathcal{M}_2 \in I, \mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2) \Rightarrow (\mathcal{M}_1 \in P \Leftrightarrow \mathcal{M}_2 \in P).$$

$\Sigma = \{a\}$  et  $P = \{\mathcal{M} \mid \mathcal{M} \text{ a un nombre pair d'états, i.e. } |Q| \text{ est pair}\}$ .

soient les machines suivantes :



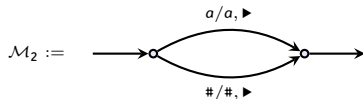
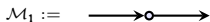
# Exemples

## Machines avec un nombre pair d'états

$$\forall \mathcal{M}_1, \mathcal{M}_2 \in I, \mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2) \Rightarrow (\mathcal{M}_1 \in P \Leftrightarrow \mathcal{M}_2 \in P).$$

$\Sigma = \{a\}$  et  $P = \{\mathcal{M} \mid \mathcal{M} \text{ a un nombre pair d'états, i.e. } |Q| \text{ est pair}\}$ .

soient les machines suivantes :



On remarque que :

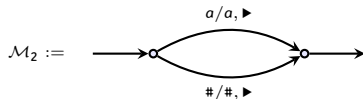
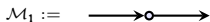
# Exemples

## Machines avec un nombre pair d'états

$$\forall \mathcal{M}_1, \mathcal{M}_2 \in I, \mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2) \Rightarrow (\mathcal{M}_1 \in P \Leftrightarrow \mathcal{M}_2 \in P).$$

$\Sigma = \{a\}$  et  $P = \{\mathcal{M} \mid \mathcal{M} \text{ a un nombre pair d'états, i.e. } |Q| \text{ est pair}\}$ .

soient les machines suivantes :



On remarque que :

1)  $\mathcal{L}(\mathcal{M}_1) = \{a\}^*$  et  $\mathcal{L}(\mathcal{M}_2) = \{a\}^*$  ;



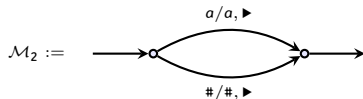
# Exemples

## Machines avec un nombre pair d'états

$$\forall \mathcal{M}_1, \mathcal{M}_2 \in I, \mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2) \Rightarrow (\mathcal{M}_1 \in P \Leftrightarrow \mathcal{M}_2 \in P).$$

$\Sigma = \{a\}$  et  $P = \{\mathcal{M} \mid \mathcal{M} \text{ a un nombre pair d'états, i.e. } |Q| \text{ est pair}\}$ .

soient les machines suivantes :



On remarque que :

- 1)  $\mathcal{L}(\mathcal{M}_1) = \{a\}^*$  et  $\mathcal{L}(\mathcal{M}_2) = \{a\}^*$  ;
- 2)  $|Q_1| = |\{0\}| = 1$  et  $|Q_2| = |\{q_0, q_1\}| = 2$  ;

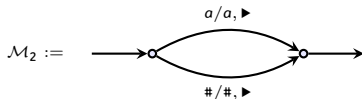
# Exemples

## Machines avec un nombre pair d'états

$$\forall \mathcal{M}_1, \mathcal{M}_2 \in I, \mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2) \Rightarrow (\mathcal{M}_1 \in P \Leftrightarrow \mathcal{M}_2 \in P).$$

$\Sigma = \{a\}$  et  $P = \{\mathcal{M} \mid \mathcal{M} \text{ a un nombre pair d'états, i.e. } |Q| \text{ est pair}\}$ .

☞ soient les machines suivantes :



☞ On remarque que :

- 1)  $\mathcal{L}(\mathcal{M}_1) = \{a\}^*$  et  $\mathcal{L}(\mathcal{M}_2) = \{a\}^*$  ;
- 2)  $|Q_1| = |\{0\}| = 1$  et  $|Q_2| = |\{q_0, q_1\}| = 2$  ;
- 3) c'est à dire que qu'on a  $\mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2)$ ,  $\mathcal{M}_1 \notin P$  et  $\mathcal{M}_2 \in P$ .

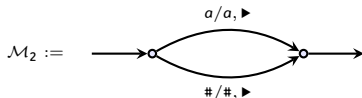
# Exemples

## Machines avec un nombre pair d'états

$$\forall \mathcal{M}_1, \mathcal{M}_2 \in I, \mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2) \Rightarrow (\mathcal{M}_1 \in P \Leftrightarrow \mathcal{M}_2 \in P).$$

$\Sigma = \{a\}$  et  $P = \{\mathcal{M} \mid \mathcal{M} \text{ a un nombre pair d'états, i.e. } |Q| \text{ est pair}\}$ .

soient les machines suivantes :



On remarque que :

- 1)  $\mathcal{L}(\mathcal{M}_1) = \{a\}^*$  et  $\mathcal{L}(\mathcal{M}_2) = \{a\}^*$  ;
- 2)  $|Q_1| = |\{0\}| = 1$  et  $|Q_2| = |\{q_0, q_1\}| = 2$  ;
- 3) c'est à dire que qu'on a  $\mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2)$ ,  $\mathcal{M}_1 \notin P$  et  $\mathcal{M}_2 \in P$ .

Cela signifie que  $P$  **n'est pas** une propriété des langages, autrement dit que le problème  $\langle \{\mathcal{M} \mid \mathcal{M} \text{ machine sur } \{a\}\}, P \rangle$  **n'est pas** un problème de langage.

$$\forall \mathcal{M}_1, \mathcal{M}_2 \in I, \mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2) \Rightarrow (\mathcal{M}_1 \in P \Leftrightarrow \mathcal{M}_2 \in P).$$

$\Sigma = \{a\}$  et  $P = \{\mathcal{M} \mid \exists n \in \mathbb{N} : a^{2n} \text{ est accepté par } \mathcal{M}\}$ .

☞ On observe l'équivalence suivante :

$$\begin{aligned} \mathcal{M} \in P &\Leftrightarrow \exists n \in \mathbb{N} : a^{2n} \text{ est accepté par } \mathcal{M} \\ &\Leftrightarrow \mathcal{L}(\mathcal{M}) \cap \{a^{2n} \mid n \in \mathbb{N}\} \neq \emptyset \end{aligned}$$

$$\forall \mathcal{M}_1, \mathcal{M}_2 \in I, \mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2) \Rightarrow (\mathcal{M}_1 \in P \Leftrightarrow \mathcal{M}_2 \in P).$$

$\Sigma = \{a\}$  et  $P = \{\mathcal{M} \mid \exists n \in \mathbb{N} : a^{2n} \text{ est accepté par } \mathcal{M}\}$ .

☞ On observe l'équivalence suivante :

$$\begin{aligned} \mathcal{M} \in P &\Leftrightarrow \exists n \in \mathbb{N} : a^{2n} \text{ est accepté par } \mathcal{M} \\ &\Leftrightarrow \mathcal{L}(\mathcal{M}) \cap \{a^{2n} \mid n \in \mathbb{N}\} \neq \emptyset \end{aligned}$$

☞ Si  $\mathcal{M}_1$  et  $\mathcal{M}_2$  sont deux machines telles que  $\mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2)$ , on a :

$$\begin{aligned} \mathcal{M}_1 \in P &\Leftrightarrow \mathcal{L}(\mathcal{M}_1) \cap \{a^{2n} \mid n \in \mathbb{N}\} \neq \emptyset \\ &\Leftrightarrow \mathcal{L}(\mathcal{M}_2) \cap \{a^{2n} \mid n \in \mathbb{N}\} \neq \emptyset \Leftrightarrow \mathcal{M}_2 \in P. \end{aligned}$$

$$\forall \mathcal{M}_1, \mathcal{M}_2 \in I, \mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2) \Rightarrow (\mathcal{M}_1 \in P \Leftrightarrow \mathcal{M}_2 \in P).$$

$\Sigma = \{a\}$  et  $P = \{\mathcal{M} \mid \exists n \in \mathbb{N} : a^{2n} \text{ est accepté par } \mathcal{M}\}$ .

☞ On observe l'équivalence suivante :

$$\begin{aligned} \mathcal{M} \in P &\Leftrightarrow \exists n \in \mathbb{N} : a^{2n} \text{ est accepté par } \mathcal{M} \\ &\Leftrightarrow \mathcal{L}(\mathcal{M}) \cap \{a^{2n} \mid n \in \mathbb{N}\} \neq \emptyset \end{aligned}$$

☞ Si  $\mathcal{M}_1$  et  $\mathcal{M}_2$  sont deux machines telles que  $\mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2)$ , on a :

$$\begin{aligned} \mathcal{M}_1 \in P &\Leftrightarrow \mathcal{L}(\mathcal{M}_1) \cap \{a^{2n} \mid n \in \mathbb{N}\} \neq \emptyset \\ &\Leftrightarrow \mathcal{L}(\mathcal{M}_2) \cap \{a^{2n} \mid n \in \mathbb{N}\} \neq \emptyset \Leftrightarrow \mathcal{M}_2 \in P. \end{aligned}$$

☞ Donc  $P$  est une propriété des langages, et  $\langle \{\mathcal{M} \mid \mathcal{M} \text{ machine sur } \{a\}\}, P \rangle$  est un problème de langage.

On va commencer par prouver un résultat un peu plus faible. Fixons un alphabet  $\Sigma$  arbitraire. On définit la machine  $\mathcal{M}_\emptyset := \langle \{q_0\}, \Sigma, \Sigma \cup \{\#\}, \emptyset, q_0, \emptyset \rangle$  :



Remarquons que  $\mathcal{L}(\mathcal{M}_\emptyset) = \emptyset$ .

lemme

Soit  $\mathcal{P} = \langle I, P \rangle$  un problème de langage des machines de Turing sur  $\Sigma$ , tel que  $\mathcal{P}$  est non-trivial et  $\mathcal{M}_\emptyset \notin P$ . Alors  $\mathcal{P}$  est indécidable.

Si  $\mathcal{P} = \langle I, P \rangle$  est un problème de langage non-trivial et si  $\mathcal{M}_\emptyset \notin P$ , alors  $\mathcal{P}$  est indécidable.

## Preuve

☞ Soit  $\mathcal{P}$  un tel problème. On va procéder par réduction depuis le problème de l'arrêt sur le mot vide :  $\mathcal{H}_\varepsilon := \{\mathcal{M} \text{ sur } \Sigma \mid \varepsilon \in \mathcal{L}(\mathcal{M})\}$ .



Si  $\mathcal{P} = \langle I, P \rangle$  est un problème de langage non-trivial et si  $\mathcal{M}_\emptyset \notin P$ , alors  $\mathcal{P}$  est indécidable.

## Preuve

- ✎ Soit  $\mathcal{P}$  un tel problème. On va procéder par réduction depuis le problème de l'arrêt sur le mot vide :  $\mathcal{H}_\varepsilon := \{\mathcal{M} \text{ sur } \Sigma \mid \varepsilon \in \mathcal{L}(\mathcal{M})\}$ .
- ✎ Pour cela, on va construire une fonction qui transforme un mot  $[\mathcal{M}]_{code}$ , avec  $\mathcal{M}$  une machine de Turing sur l'alphabet  $\Sigma$ , en un mot  $[\mathcal{M}']_{code}$ , telle que :

Si  $\mathcal{P} = \langle I, P \rangle$  est un problème de langage non-trivial et si  $\mathcal{M}_\emptyset \notin P$ , alors  $\mathcal{P}$  est indécidable.

## Preuve

- ✎ Soit  $\mathcal{P}$  un tel problème. On va procéder par réduction depuis le problème de l'arrêt sur le mot vide :  $\mathcal{H}_\varepsilon := \{\mathcal{M} \text{ sur } \Sigma \mid \varepsilon \in \mathcal{L}(\mathcal{M})\}$ .
- ✎ Pour cela, on va construire une fonction qui transforme un mot  $[\mathcal{M}]_{code}$ , avec  $\mathcal{M}$  une machine de Turing sur l'alphabet  $\Sigma$ , en un mot  $[\mathcal{M}']_{code}$ , telle que :
  - $\mathcal{M}'$  est une machine de Turing sur l'alphabet  $\Sigma$ ,

Si  $\mathcal{P} = \langle I, P \rangle$  est un problème de langage non-trivial et si  $\mathcal{M}_\emptyset \notin P$ , alors  $\mathcal{P}$  est indécidable.

## Preuve

- ✎ Soit  $\mathcal{P}$  un tel problème. On va procéder par réduction depuis le problème de l'arrêt sur le mot vide :  $\mathcal{H}_\varepsilon := \{\mathcal{M} \text{ sur } \Sigma \mid \varepsilon \in \mathcal{L}(\mathcal{M})\}$ .
- ✎ Pour cela, on va construire une fonction qui transforme un mot  $[\mathcal{M}]_{code}$ , avec  $\mathcal{M}$  une machine de Turing sur l'alphabet  $\Sigma$ , en un mot  $[\mathcal{M}']_{code}$ , telle que :
  - $\mathcal{M}'$  est une machine de Turing sur l'alphabet  $\Sigma$ ,
  - $\varepsilon \in \mathcal{L}(\mathcal{M}) \Leftrightarrow \mathcal{M}' \in P$ ,

Si  $\mathcal{P} = \langle I, P \rangle$  est un problème de langage non-trivial et si  $\mathcal{M}_\emptyset \notin P$ , alors  $\mathcal{P}$  est indécidable.

## Preuve

- ✎ Soit  $\mathcal{P}$  un tel problème. On va procéder par réduction depuis le problème de l'arrêt sur le mot vide :  $\mathcal{H}_\varepsilon := \{\mathcal{M} \text{ sur } \Sigma \mid \varepsilon \in \mathcal{L}(\mathcal{M})\}$ .
- ✎ Pour cela, on va construire une fonction qui transforme un mot  $[\mathcal{M}]_{code}$ , avec  $\mathcal{M}$  une machine de Turing sur l'alphabet  $\Sigma$ , en un mot  $[\mathcal{M}']_{code}$ , telle que :
  - $\mathcal{M}'$  est une machine de Turing sur l'alphabet  $\Sigma$ ,
  - $\varepsilon \in \mathcal{L}(\mathcal{M}) \Leftrightarrow \mathcal{M}' \in P$ ,
  - la fonction est calculable.

Si  $\mathcal{P} = \langle I, P \rangle$  est un problème de langage non-trivial et si  $\mathcal{M}_\emptyset \notin P$ , alors  $\mathcal{P}$  est indécidable.

## Preuve

- ✎ Soit  $\mathcal{P}$  un tel problème. On va procéder par réduction depuis le problème de l'arrêt sur le mot vide :  $\mathcal{H}_\varepsilon := \{\mathcal{M} \text{ sur } \Sigma \mid \varepsilon \in \mathcal{L}(\mathcal{M})\}$ .
- ✎ Pour cela, on va construire une fonction qui transforme un mot  $[\mathcal{M}]_{code}$ , avec  $\mathcal{M}$  une machine de Turing sur l'alphabet  $\Sigma$ , en un mot  $[\mathcal{M}']_{code}$ , telle que :
  - $\mathcal{M}'$  est une machine de Turing sur l'alphabet  $\Sigma$ ,
  - $\varepsilon \in \mathcal{L}(\mathcal{M}) \Leftrightarrow \mathcal{M}' \in P$ ,
  - la fonction est calculable.
- ✎ En fait, on va décrire la machine  $\mathcal{M}'$  produite à partir d'une machine d'entrée  $\mathcal{M}$ , et on va se convaincre que l'on pourrait réaliser cette construction à partir du code de  $\mathcal{M}$ .

Si  $\mathcal{P} = \langle I, P \rangle$  est un problème de langage non-trivial et si  $\mathcal{M}_\emptyset \notin P$ , alors  $\mathcal{P}$  est indécidable.

## Preuve

- ✎ Soit  $\mathcal{P}$  un tel problème. On va procéder par réduction depuis le problème de l'arrêt sur le mot vide :  $\mathcal{H}_\varepsilon := \{\mathcal{M} \text{ sur } \Sigma \mid \varepsilon \in \mathcal{L}(\mathcal{M})\}$ .
- ✎ Pour cela, on va construire une fonction qui transforme un mot  $[\mathcal{M}]_{code}$ , avec  $\mathcal{M}$  une machine de Turing sur l'alphabet  $\Sigma$ , en un mot  $[\mathcal{M}']_{code}$ , telle que :
  - $\mathcal{M}'$  est une machine de Turing sur l'alphabet  $\Sigma$ ,
  - $\varepsilon \in \mathcal{L}(\mathcal{M}) \Leftrightarrow \mathcal{M}' \in P$ ,
  - la fonction est calculable.
- ✎ En fait, on va décrire la machine  $\mathcal{M}'$  produite à partir d'une machine d'entrée  $\mathcal{M}$ , et on va se convaincre que l'on pourrait réaliser cette construction à partir du code de  $\mathcal{M}$ .
- ✎ Comme  $\mathcal{M}_\emptyset \notin P$ , et comme  $\mathcal{P}$  est non-trivial, alors il existe une machine  $\mathcal{M}_{oui} \in P$ .

Si  $\mathcal{P} = \langle I, P \rangle$  est un problème de langage non-trivial et si  $\mathcal{M}_\emptyset \notin P$ , alors  $\mathcal{P}$  est indécidable.

## Preuve

- ✎ Soit  $\mathcal{P}$  un tel problème. On va procéder par réduction depuis le problème de l'arrêt sur le mot vide :  $\mathcal{H}_\varepsilon := \{\mathcal{M} \text{ sur } \Sigma \mid \varepsilon \in \mathcal{L}(\mathcal{M})\}$ .
- ✎ Pour cela, on va construire une fonction qui transforme un mot  $[\mathcal{M}]_{code}$ , avec  $\mathcal{M}$  une machine de Turing sur l'alphabet  $\Sigma$ , en un mot  $[\mathcal{M}']_{code}$ , telle que :
  - $\mathcal{M}'$  est une machine de Turing sur l'alphabet  $\Sigma$ ,
  - $\varepsilon \in \mathcal{L}(\mathcal{M}) \Leftrightarrow \mathcal{M}' \in P$ ,
  - la fonction est calculable.
- ✎ En fait, on va décrire la machine  $\mathcal{M}'$  produite à partir d'une machine d'entrée  $\mathcal{M}$ , et on va se convaincre que l'on pourrait réaliser cette construction à partir du code de  $\mathcal{M}$ .
- ✎ Comme  $\mathcal{M}_\emptyset \notin P$ , et comme  $\mathcal{P}$  est non-trivial, alors il existe une machine  $\mathcal{M}_{oui} \in P$ .
- ✎ Soit en entrée une machine avec un seul ruban  $\mathcal{M}$ . On construit la machine  $\mathcal{M}'$  avec deux rubans, et qui reçoit son entrée sur le premier ruban :

Si  $\mathcal{P} = \langle I, P \rangle$  est un problème de langage non-trivial et si  $\mathcal{M}_\emptyset \notin P$ , alors  $\mathcal{P}$  est indécidable.

## Preuve

- ✎ Soit  $\mathcal{P}$  un tel problème. On va procéder par réduction depuis le problème de l'arrêt sur le mot vide :  $\mathcal{H}_\varepsilon := \{\mathcal{M} \text{ sur } \Sigma \mid \varepsilon \in \mathcal{L}(\mathcal{M})\}$ .
- ✎ Pour cela, on va construire une fonction qui transforme un mot  $[\mathcal{M}]_{code}$ , avec  $\mathcal{M}$  une machine de Turing sur l'alphabet  $\Sigma$ , en un mot  $[\mathcal{M}']_{code}$ , telle que :
  - $\mathcal{M}'$  est une machine de Turing sur l'alphabet  $\Sigma$ ,
  - $\varepsilon \in \mathcal{L}(\mathcal{M}) \Leftrightarrow \mathcal{M}' \in P$ ,
  - la fonction est calculable.
- ✎ En fait, on va décrire la machine  $\mathcal{M}'$  produite à partir d'une machine d'entrée  $\mathcal{M}$ , et on va se convaincre que l'on pourrait réaliser cette construction à partir du code de  $\mathcal{M}$ .
- ✎ Comme  $\mathcal{M}_\emptyset \notin P$ , et comme  $\mathcal{P}$  est non-trivial, alors il existe une machine  $\mathcal{M}_{oui} \in P$ .
- ✎ Soit en entrée une machine avec un seul ruban  $\mathcal{M}$ . On construit la machine  $\mathcal{M}'$  avec deux rubans, et qui reçoit son entrée sur le premier ruban :
  - 1) avant toute inspection du premier ruban,  $\mathcal{M}'$  utilise son second ruban (initialement vide) pour exécuter  $\mathcal{M}$



Si  $\mathcal{P} = \langle I, P \rangle$  est un problème de langage non-trivial et si  $\mathcal{M}_\emptyset \notin P$ , alors  $\mathcal{P}$  est indécidable.

## Preuve

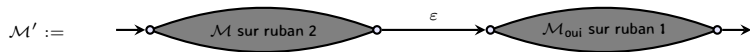
- ✎ Soit  $\mathcal{P}$  un tel problème. On va procéder par réduction depuis le problème de l'arrêt sur le mot vide :  $\mathcal{H}_\varepsilon := \{\mathcal{M} \text{ sur } \Sigma \mid \varepsilon \in \mathcal{L}(\mathcal{M})\}$ .
- ✎ Pour cela, on va construire une fonction qui transforme un mot  $[\mathcal{M}]_{code}$ , avec  $\mathcal{M}$  une machine de Turing sur l'alphabet  $\Sigma$ , en un mot  $[\mathcal{M}']_{code}$ , telle que :
  - $\mathcal{M}'$  est une machine de Turing sur l'alphabet  $\Sigma$ ,
  - $\varepsilon \in \mathcal{L}(\mathcal{M}) \Leftrightarrow \mathcal{M}' \in P$ ,
  - la fonction est calculable.
- ✎ En fait, on va décrire la machine  $\mathcal{M}'$  produite à partir d'une machine d'entrée  $\mathcal{M}$ , et on va se convaincre que l'on pourrait réaliser cette construction à partir du code de  $\mathcal{M}$ .
- ✎ Comme  $\mathcal{M}_\emptyset \notin P$ , et comme  $\mathcal{P}$  est non-trivial, alors il existe une machine  $\mathcal{M}_{oui} \in P$ .
- ✎ Soit en entrée une machine avec un seul ruban  $\mathcal{M}$ . On construit la machine  $\mathcal{M}'$  avec deux rubans, et qui reçoit son entrée sur le premier ruban :
  - 1) avant toute inspection du premier ruban,  $\mathcal{M}'$  utilise son second ruban (initialement vide) pour exécuter  $\mathcal{M}$
  - 2) Si cette exécution se termine sur un état acceptant,  $\mathcal{M}'$  exécute alors  $\mathcal{M}_{oui}$  sur son premier ruban.

Si  $\mathcal{P} = \langle I, P \rangle$  est un problème de langage non-trivial et si  $\mathcal{M}_\emptyset \notin P$ , alors  $\mathcal{P}$  est indécidable.

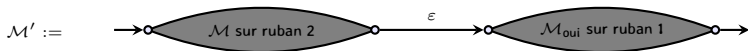
## Preuve

- ✎ Soit  $\mathcal{P}$  un tel problème. On va procéder par réduction depuis le problème de l'arrêt sur le mot vide :  $\mathcal{H}_\varepsilon := \{\mathcal{M} \text{ sur } \Sigma \mid \varepsilon \in \mathcal{L}(\mathcal{M})\}$ .
- ✎ Pour cela, on va construire une fonction qui transforme un mot  $[\mathcal{M}]_{code}$ , avec  $\mathcal{M}$  une machine de Turing sur l'alphabet  $\Sigma$ , en un mot  $[\mathcal{M}']_{code}$ , telle que :
  - $\mathcal{M}'$  est une machine de Turing sur l'alphabet  $\Sigma$ ,
  - $\varepsilon \in \mathcal{L}(\mathcal{M}) \Leftrightarrow \mathcal{M}' \in P$ ,
  - la fonction est calculable.
- ✎ En fait, on va décrire la machine  $\mathcal{M}'$  produite à partir d'une machine d'entrée  $\mathcal{M}$ , et on va se convaincre que l'on pourrait réaliser cette construction à partir du code de  $\mathcal{M}$ .
- ✎ Comme  $\mathcal{M}_\emptyset \notin P$ , et comme  $\mathcal{P}$  est non-trivial, alors il existe une machine  $\mathcal{M}_{oui} \in P$ .
- ✎ Soit en entrée une machine avec un seul ruban  $\mathcal{M}$ . On construit la machine  $\mathcal{M}'$  avec deux rubans, et qui reçoit son entrée sur le premier ruban :
  - 1) avant toute inspection du premier ruban,  $\mathcal{M}'$  utilise son second ruban (initialement vide) pour exécuter  $\mathcal{M}$
  - 2) Si cette exécution se termine sur un état acceptant,  $\mathcal{M}'$  exécute alors  $\mathcal{M}_{oui}$  sur son premier ruban.
  - 3) Les états acceptants de  $\mathcal{M}'$  sont ceux de  $\mathcal{M}_{oui}$ .

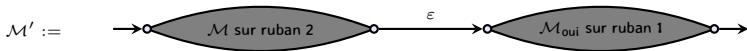
## Preuve du théorème (suite)



👉 Construction de  $[\mathcal{M}']_{code}$  à partir de code  $[\mathcal{M}]_{code}$  (et de  $[\mathcal{M}_{oui}]_{code}$ ) :



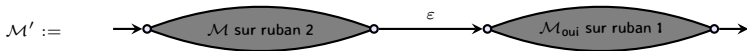
- ☞ Construction de  $[\mathcal{M}']_{code}$  à partir de code  $[\mathcal{M}]_{code}$  (et de  $[\mathcal{M}_{\text{oui}}]_{code}$ ) :
- Renommer les états de  $\mathcal{M}_{\text{oui}}$  pour que  $Q_{\mathcal{M}} \cap Q_{\mathcal{M}_{\text{oui}}} = \emptyset$ .



Construction de  $[\mathcal{M}']_{code}$  à partir de code  $[\mathcal{M}]_{code}$  (et de  $[\mathcal{M}_{\text{oui}}]_{code}$ ) :

- Renommer les états de  $\mathcal{M}_{\text{oui}}$  pour que  $Q_{\mathcal{M}} \cap Q_{\mathcal{M}_{\text{oui}}} = \emptyset$ .

- Transformer les transitions  $p \xrightarrow{a/b, M} q$  en transitions  $p \xrightarrow{\# \begin{matrix} a \\ \downarrow \end{matrix} / \# \begin{matrix} b \\ \downarrow \end{matrix}, M} q$ .

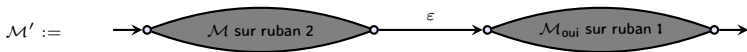


👉 Construction de  $[\mathcal{M}']_{code}$  à partir de code  $[\mathcal{M}]_{code}$  (et de  $[\mathcal{M}_{oui}]_{code}$ ) :

- Renommer les états de  $\mathcal{M}_{oui}$  pour que  $Q_{\mathcal{M}} \cap Q_{\mathcal{M}_{oui}} = \emptyset$ .

- Transformer les transitions  $p \xrightarrow{a/b, M} q$  en transitions  $p \xrightarrow{\# \overset{\blacktriangledown}{a} / \# \overset{\blacktriangledown}{b}, M} q$ .

- Transformer les transitions  $p \xrightarrow{a/b, M} q$  en transitions  $p \xrightarrow{\overset{\blacktriangledown}{a} / \# \overset{\blacktriangledown}{b}, M} q$ .



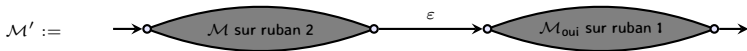
☞ Construction de  $[\mathcal{M}']_{code}$  à partir de code  $[\mathcal{M}]_{code}$  (et de  $[\mathcal{M}_{oui}]_{code}$ ) :

- Renommer les états de  $\mathcal{M}_{oui}$  pour que  $Q_{\mathcal{M}} \cap Q_{\mathcal{M}_{oui}} = \emptyset$ .

- Transformer les transitions  $p \xrightarrow{a/b, M} q$  en transitions  $p \xrightarrow{\# \overset{\blacktriangledown}{a} / \# \overset{\blacktriangledown}{b}, M} q$ .

- Transformer les transitions  $p \xrightarrow{a/b, M} q$  en transitions  $p \xrightarrow{\# \overset{\blacktriangledown}{a} / \# \overset{\blacktriangledown}{b}, M} q$ .

- Ajouter des transitions  $q_f \xrightarrow{\# \overset{\blacktriangledown}{a} / \# \overset{\blacktriangledown}{a'}} q_0^{oui}$  pour tout  $q_f \in F_{\mathcal{M}}$ .



👉 Construction de  $[\mathcal{M}']_{code}$  à partir de code  $[\mathcal{M}]_{code}$  (et de  $[\mathcal{M}_{oui}]_{code}$ ) :

- Renommer les états de  $\mathcal{M}_{oui}$  pour que  $Q_{\mathcal{M}} \cap Q_{\mathcal{M}_{oui}} = \emptyset$ .

- Transformer les transitions  $p \xrightarrow{a/b, M} q$  en transitions  $p \xrightarrow{\# \overset{\blacktriangledown}{a} / \# \overset{\blacktriangledown}{b}, M} q$ .

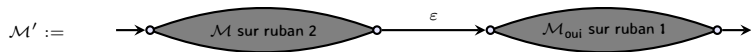
- Transformer les transitions  $p \xrightarrow{a/b, M} q$  en transitions  $p \xrightarrow{\# \overset{\blacktriangledown}{a} / \# \overset{\blacktriangledown}{b}, M} q$ .

- Ajouter des transitions  $q_f \xrightarrow{\# \overset{\blacktriangledown}{a} / \# \overset{\blacktriangledown}{a}, \blacktriangledown} q_0^{oui}$  pour tout  $q_f \in F_{\mathcal{M}}$ .

👉 Il est donc raisonnable de considérer que cette fonction est calculable.

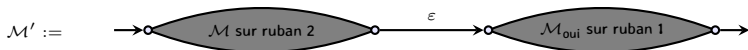


## Preuve du théorème (suite)



👉 Analysons le comportement de  $\mathcal{M}'$  sur une entrée  $w \in \Sigma^*$  :

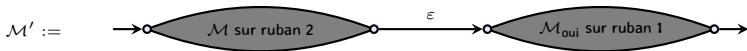
## Preuve du théorème (suite)



👉 Analysons le comportement de  $\mathcal{M}'$  sur une entrée  $w \in \Sigma^*$  :

- Si  $\varepsilon \in \mathcal{L}(\mathcal{M})$ ,  
alors l'étape 1 se termine dans un état acceptant de  $\mathcal{M}$ , et on passe à l'étape 2.  
Dans ce cas,  $w \in \mathcal{L}(\mathcal{M}') \Leftrightarrow w \in \mathcal{L}(\mathcal{M}_{oui})$ .

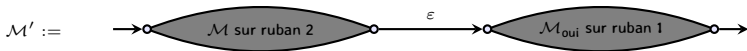
## Preuve du théorème (suite)



👉 Analysons le comportement de  $\mathcal{M}'$  sur une entrée  $w \in \Sigma^*$  :

- Si  $\varepsilon \in \mathcal{L}(\mathcal{M})$ ,  
alors l'étape 1 se termine dans un état acceptant de  $\mathcal{M}$ , et on passe à l'étape 2.  
Dans ce cas,  $w \in \mathcal{L}(\mathcal{M}') \Leftrightarrow w \in \mathcal{L}(\mathcal{M}_{\text{oui}})$ .
- Si  $\varepsilon \notin \mathcal{L}(\mathcal{M})$ ,  
alors soit l'étape 1 se bloque sur un état non-acceptant, soit elle boucle.  
Dans les deux cas, on évite l'étape 2, et on ne visite jamais d'état acceptant.  
Donc  $w \notin \mathcal{L}(\mathcal{M}')$ .

## Preuve du théorème (suite)



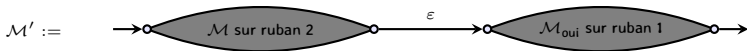
👉 Analysons le comportement de  $\mathcal{M}'$  sur une entrée  $w \in \Sigma^*$  :

- Si  $\varepsilon \in \mathcal{L}(\mathcal{M})$ ,  
alors l'étape 1 se termine dans un état acceptant de  $\mathcal{M}$ , et on passe à l'étape 2.  
Dans ce cas,  $w \in \mathcal{L}(\mathcal{M}') \Leftrightarrow w \in \mathcal{L}(\mathcal{M}_{\text{oui}})$ .
- Si  $\varepsilon \notin \mathcal{L}(\mathcal{M})$ ,  
alors soit l'étape 1 se bloque sur un état non-acceptant, soit elle boucle.  
Dans les deux cas, on évite l'étape 2, et on ne visite jamais d'état acceptant.  
Donc  $w \notin \mathcal{L}(\mathcal{M}')$ .

👉 Autrement dit :

$$\mathcal{M} \in \mathcal{H}_\varepsilon$$

## Preuve du théorème (suite)

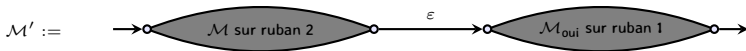


👉 Analysons le comportement de  $\mathcal{M}'$  sur une entrée  $w \in \Sigma^*$  :

- Si  $\varepsilon \in \mathcal{L}(\mathcal{M})$ ,  
alors l'étape 1 se termine dans un état acceptant de  $\mathcal{M}$ , et on passe à l'étape 2.  
Dans ce cas,  $w \in \mathcal{L}(\mathcal{M}') \Leftrightarrow w \in \mathcal{L}(\mathcal{M}_{\text{oui}})$ .
- Si  $\varepsilon \notin \mathcal{L}(\mathcal{M})$ ,  
alors soit l'étape 1 se bloque sur un état non-acceptant, soit elle boucle.  
Dans les deux cas, on évite l'étape 2, et on ne visite jamais d'état acceptant.  
Donc  $w \notin \mathcal{L}(\mathcal{M}')$ .

👉 Autrement dit :

$$\mathcal{M} \in \mathcal{H}_\varepsilon \Leftrightarrow \varepsilon \in \mathcal{L}(\mathcal{M})$$

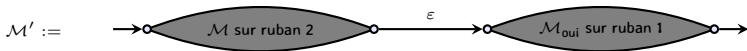


👉 Analysons le comportement de  $\mathcal{M}'$  sur une entrée  $w \in \Sigma^*$  :

- Si  $\varepsilon \in \mathcal{L}(\mathcal{M})$ ,  
alors l'étape 1 se termine dans un état acceptant de  $\mathcal{M}$ , et on passe à l'étape 2.  
Dans ce cas,  $w \in \mathcal{L}(\mathcal{M}') \Leftrightarrow w \in \mathcal{L}(\mathcal{M}_{\text{oui}})$ .
- Si  $\varepsilon \notin \mathcal{L}(\mathcal{M})$ ,  
alors soit l'étape 1 se bloque sur un état non-acceptant, soit elle boucle.  
Dans les deux cas, on évite l'étape 2, et on ne visite jamais d'état acceptant.  
Donc  $w \notin \mathcal{L}(\mathcal{M}')$ .

👉 Autrement dit :

$$\mathcal{M} \in \mathcal{H}_\varepsilon \Leftrightarrow \varepsilon \in \mathcal{L}(\mathcal{M}) \Rightarrow (\forall w \in \Sigma^*, w \in \mathcal{L}(\mathcal{M}') \Leftrightarrow w \in \mathcal{L}(\mathcal{M}_{\text{oui}}))$$

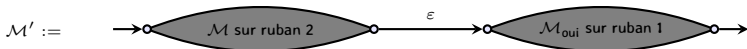


👉 Analysons le comportement de  $\mathcal{M}'$  sur une entrée  $w \in \Sigma^*$  :

- Si  $\varepsilon \in \mathcal{L}(\mathcal{M})$ ,  
alors l'étape 1 se termine dans un état acceptant de  $\mathcal{M}$ , et on passe à l'étape 2.  
Dans ce cas,  $w \in \mathcal{L}(\mathcal{M}') \Leftrightarrow w \in \mathcal{L}(\mathcal{M}_{\text{oui}})$ .
- Si  $\varepsilon \notin \mathcal{L}(\mathcal{M})$ ,  
alors soit l'étape 1 se bloque sur un état non-acceptant, soit elle boucle.  
Dans les deux cas, on évite l'étape 2, et on ne visite jamais d'état acceptant.  
Donc  $w \notin \mathcal{L}(\mathcal{M}')$ .

👉 Autrement dit :

$$\begin{aligned} \mathcal{M} \in \mathcal{H}_\varepsilon \Leftrightarrow \varepsilon \in \mathcal{L}(\mathcal{M}) &\Rightarrow (\forall w \in \Sigma^*, w \in \mathcal{L}(\mathcal{M}') \Leftrightarrow w \in \mathcal{L}(\mathcal{M}_{\text{oui}})) \\ &\Leftrightarrow \mathcal{L}(\mathcal{M}') = \mathcal{L}(\mathcal{M}_{\text{oui}}) \end{aligned}$$



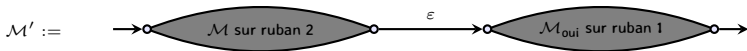
👉 Analysons le comportement de  $\mathcal{M}'$  sur une entrée  $w \in \Sigma^*$  :

- Si  $\varepsilon \in \mathcal{L}(\mathcal{M})$ ,  
alors l'étape 1 se termine dans un état acceptant de  $\mathcal{M}$ , et on passe à l'étape 2.  
Dans ce cas,  $w \in \mathcal{L}(\mathcal{M}') \Leftrightarrow w \in \mathcal{L}(\mathcal{M}_{\text{oui}})$ .
- Si  $\varepsilon \notin \mathcal{L}(\mathcal{M})$ ,  
alors soit l'étape 1 se bloque sur un état non-acceptant, soit elle boucle.  
Dans les deux cas, on évite l'étape 2, et on ne visite jamais d'état acceptant.  
Donc  $w \notin \mathcal{L}(\mathcal{M}')$ .

👉 Autrement dit :

$$\begin{aligned}\mathcal{M} \in \mathcal{H}_\varepsilon &\Leftrightarrow \varepsilon \in \mathcal{L}(\mathcal{M}) \Rightarrow (\forall w \in \Sigma^*, w \in \mathcal{L}(\mathcal{M}') \Leftrightarrow w \in \mathcal{L}(\mathcal{M}_{\text{oui}})) \\ &\Leftrightarrow \mathcal{L}(\mathcal{M}') = \mathcal{L}(\mathcal{M}_{\text{oui}}) \\ &\Rightarrow \mathcal{M}' \in \mathcal{P}\end{aligned}$$





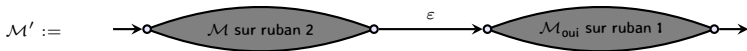
👉 Analysons le comportement de  $\mathcal{M}'$  sur une entrée  $w \in \Sigma^*$  :

- Si  $\varepsilon \in \mathcal{L}(\mathcal{M})$ ,  
alors l'étape 1 se termine dans un état acceptant de  $\mathcal{M}$ , et on passe à l'étape 2.  
Dans ce cas,  $w \in \mathcal{L}(\mathcal{M}') \Leftrightarrow w \in \mathcal{L}(\mathcal{M}_{\text{oui}})$ .
- Si  $\varepsilon \notin \mathcal{L}(\mathcal{M})$ ,  
alors soit l'étape 1 se bloque sur un état non-acceptant, soit elle boucle.  
Dans les deux cas, on évite l'étape 2, et on ne visite jamais d'état acceptant.  
Donc  $w \notin \mathcal{L}(\mathcal{M}')$ .

👉 Autrement dit :

$$\begin{aligned} \mathcal{M} \in \mathcal{H}_\varepsilon &\Leftrightarrow \varepsilon \in \mathcal{L}(\mathcal{M}) \Rightarrow (\forall w \in \Sigma^*, w \in \mathcal{L}(\mathcal{M}') \Leftrightarrow w \in \mathcal{L}(\mathcal{M}_{\text{oui}})) \\ &\Leftrightarrow \mathcal{L}(\mathcal{M}') = \mathcal{L}(\mathcal{M}_{\text{oui}}) \\ &\Rightarrow \mathcal{M}' \in \mathcal{P} \end{aligned}$$

$$\mathcal{M} \notin \mathcal{H}_\varepsilon$$



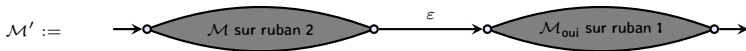
👉 Analysons le comportement de  $\mathcal{M}'$  sur une entrée  $w \in \Sigma^*$  :

- Si  $\varepsilon \in \mathcal{L}(\mathcal{M})$ ,  
alors l'étape 1 se termine dans un état acceptant de  $\mathcal{M}$ , et on passe à l'étape 2.  
Dans ce cas,  $w \in \mathcal{L}(\mathcal{M}') \Leftrightarrow w \in \mathcal{L}(\mathcal{M}_{\text{oui}})$ .
- Si  $\varepsilon \notin \mathcal{L}(\mathcal{M})$ ,  
alors soit l'étape 1 se bloque sur un état non-acceptant, soit elle boucle.  
Dans les deux cas, on évite l'étape 2, et on ne visite jamais d'état acceptant.  
Donc  $w \notin \mathcal{L}(\mathcal{M}')$ .

👉 Autrement dit :

$$\begin{aligned} \mathcal{M} \in \mathcal{H}_\varepsilon &\Leftrightarrow \varepsilon \in \mathcal{L}(\mathcal{M}) \Rightarrow (\forall w \in \Sigma^*, w \in \mathcal{L}(\mathcal{M}') \Leftrightarrow w \in \mathcal{L}(\mathcal{M}_{\text{oui}})) \\ &\Leftrightarrow \mathcal{L}(\mathcal{M}') = \mathcal{L}(\mathcal{M}_{\text{oui}}) \\ &\Rightarrow \mathcal{M}' \in \mathcal{P} \end{aligned}$$

$$\mathcal{M} \notin \mathcal{H}_\varepsilon \Leftrightarrow \varepsilon \notin \mathcal{L}(\mathcal{M})$$



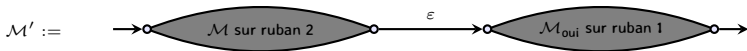
👉 Analysons le comportement de  $\mathcal{M}'$  sur une entrée  $w \in \Sigma^*$  :

- Si  $\varepsilon \in \mathcal{L}(\mathcal{M})$ ,  
alors l'étape 1 se termine dans un état acceptant de  $\mathcal{M}$ , et on passe à l'étape 2.  
Dans ce cas,  $w \in \mathcal{L}(\mathcal{M}') \Leftrightarrow w \in \mathcal{L}(\mathcal{M}_{\text{oui}})$ .
- Si  $\varepsilon \notin \mathcal{L}(\mathcal{M})$ ,  
alors soit l'étape 1 se bloque sur un état non-acceptant, soit elle boucle.  
Dans les deux cas, on évite l'étape 2, et on ne visite jamais d'état acceptant.  
Donc  $w \notin \mathcal{L}(\mathcal{M}')$ .

👉 Autrement dit :

$$\begin{aligned} \mathcal{M} \in \mathcal{H}_\varepsilon &\Leftrightarrow \varepsilon \in \mathcal{L}(\mathcal{M}) \Rightarrow (\forall w \in \Sigma^*, w \in \mathcal{L}(\mathcal{M}') \Leftrightarrow w \in \mathcal{L}(\mathcal{M}_{\text{oui}})) \\ &\Leftrightarrow \mathcal{L}(\mathcal{M}') = \mathcal{L}(\mathcal{M}_{\text{oui}}) \\ &\Rightarrow \mathcal{M}' \in \mathcal{P} \end{aligned}$$

$$\mathcal{M} \notin \mathcal{H}_\varepsilon \Leftrightarrow \varepsilon \notin \mathcal{L}(\mathcal{M}) \Rightarrow \forall w \in \Sigma^*, w \notin \mathcal{L}(\mathcal{M}')$$



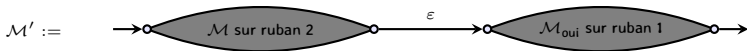
👉 Analysons le comportement de  $\mathcal{M}'$  sur une entrée  $w \in \Sigma^*$  :

- Si  $\varepsilon \in \mathcal{L}(\mathcal{M})$ ,  
alors l'étape 1 se termine dans un état acceptant de  $\mathcal{M}$ , et on passe à l'étape 2.  
Dans ce cas,  $w \in \mathcal{L}(\mathcal{M}') \Leftrightarrow w \in \mathcal{L}(\mathcal{M}_{\text{oui}})$ .
- Si  $\varepsilon \notin \mathcal{L}(\mathcal{M})$ ,  
alors soit l'étape 1 se bloque sur un état non-acceptant, soit elle boucle.  
Dans les deux cas, on évite l'étape 2, et on ne visite jamais d'état acceptant.  
Donc  $w \notin \mathcal{L}(\mathcal{M}')$ .

👉 Autrement dit :

$$\begin{aligned} \mathcal{M} \in \mathcal{H}_\varepsilon &\Leftrightarrow \varepsilon \in \mathcal{L}(\mathcal{M}) \Rightarrow (\forall w \in \Sigma^*, w \in \mathcal{L}(\mathcal{M}') \Leftrightarrow w \in \mathcal{L}(\mathcal{M}_{\text{oui}})) \\ &\Leftrightarrow \mathcal{L}(\mathcal{M}') = \mathcal{L}(\mathcal{M}_{\text{oui}}) \\ &\Rightarrow \mathcal{M}' \in \mathcal{P} \end{aligned}$$

$$\begin{aligned} \mathcal{M} \notin \mathcal{H}_\varepsilon &\Leftrightarrow \varepsilon \notin \mathcal{L}(\mathcal{M}) \Rightarrow \forall w \in \Sigma^*, w \notin \mathcal{L}(\mathcal{M}') \\ &\Leftrightarrow \mathcal{L}(\mathcal{M}') = \emptyset = \mathcal{L}(\mathcal{M}_\emptyset) \end{aligned}$$



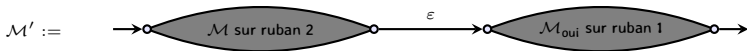
👉 Analysons le comportement de  $\mathcal{M}'$  sur une entrée  $w \in \Sigma^*$  :

- Si  $\varepsilon \in \mathcal{L}(\mathcal{M})$ ,  
alors l'étape 1 se termine dans un état acceptant de  $\mathcal{M}$ , et on passe à l'étape 2.  
Dans ce cas,  $w \in \mathcal{L}(\mathcal{M}') \Leftrightarrow w \in \mathcal{L}(\mathcal{M}_{\text{oui}})$ .
- Si  $\varepsilon \notin \mathcal{L}(\mathcal{M})$ ,  
alors soit l'étape 1 se bloque sur un état non-acceptant, soit elle boucle.  
Dans les deux cas, on évite l'étape 2, et on ne visite jamais d'état acceptant.  
Donc  $w \notin \mathcal{L}(\mathcal{M}')$ .

👉 Autrement dit :

$$\begin{aligned} \mathcal{M} \in \mathcal{H}_\varepsilon &\Leftrightarrow \varepsilon \in \mathcal{L}(\mathcal{M}) \Rightarrow (\forall w \in \Sigma^*, w \in \mathcal{L}(\mathcal{M}') \Leftrightarrow w \in \mathcal{L}(\mathcal{M}_{\text{oui}})) \\ &\Leftrightarrow \mathcal{L}(\mathcal{M}') = \mathcal{L}(\mathcal{M}_{\text{oui}}) \\ &\Rightarrow \mathcal{M}' \in \mathcal{P} \end{aligned}$$

$$\begin{aligned} \mathcal{M} \notin \mathcal{H}_\varepsilon &\Leftrightarrow \varepsilon \notin \mathcal{L}(\mathcal{M}) \Rightarrow \forall w \in \Sigma^*, w \notin \mathcal{L}(\mathcal{M}') \\ &\Leftrightarrow \mathcal{L}(\mathcal{M}') = \emptyset = \mathcal{L}(\mathcal{M}_\emptyset) \\ &\Rightarrow \mathcal{M}' \notin \mathcal{P} \end{aligned}$$



👉 Analysons le comportement de  $\mathcal{M}'$  sur une entrée  $w \in \Sigma^*$  :

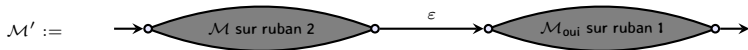
- Si  $\varepsilon \in \mathcal{L}(\mathcal{M})$ ,  
alors l'étape 1 se termine dans un état acceptant de  $\mathcal{M}$ , et on passe à l'étape 2.  
Dans ce cas,  $w \in \mathcal{L}(\mathcal{M}') \Leftrightarrow w \in \mathcal{L}(\mathcal{M}_{\text{oui}})$ .
- Si  $\varepsilon \notin \mathcal{L}(\mathcal{M})$ ,  
alors soit l'étape 1 se bloque sur un état non-acceptant, soit elle boucle.  
Dans les deux cas, on évite l'étape 2, et on ne visite jamais d'état acceptant.  
Donc  $w \notin \mathcal{L}(\mathcal{M}')$ .

👉 Autrement dit :

$$\begin{aligned} \mathcal{M} \in \mathcal{H}_\varepsilon &\Leftrightarrow \varepsilon \in \mathcal{L}(\mathcal{M}) \Rightarrow (\forall w \in \Sigma^*, w \in \mathcal{L}(\mathcal{M}') \Leftrightarrow w \in \mathcal{L}(\mathcal{M}_{\text{oui}})) \\ &\Leftrightarrow \mathcal{L}(\mathcal{M}') = \mathcal{L}(\mathcal{M}_{\text{oui}}) \\ &\Rightarrow \mathcal{M}' \in \mathcal{P} \end{aligned}$$

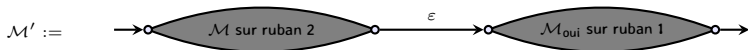
$$\begin{aligned} \mathcal{M} \notin \mathcal{H}_\varepsilon &\Leftrightarrow \varepsilon \notin \mathcal{L}(\mathcal{M}) \Rightarrow \forall w \in \Sigma^*, w \notin \mathcal{L}(\mathcal{M}') \\ &\Leftrightarrow \mathcal{L}(\mathcal{M}') = \emptyset = \mathcal{L}(\mathcal{M}_\emptyset) \\ &\Rightarrow \mathcal{M}' \notin \mathcal{P} \end{aligned}$$

$$\mathcal{M} \in \mathcal{H}_\varepsilon \Leftrightarrow \mathcal{M}' \in \mathcal{P}$$



👉 On a donc une fonction  $\rho : [\mathcal{M}]_{code} \mapsto [\mathcal{M}']_{code}$  **calculable** telle que

$$\forall \mathcal{M}, \mathcal{M} \in \mathcal{H}_\epsilon \Leftrightarrow \mathcal{M}' \in P.$$

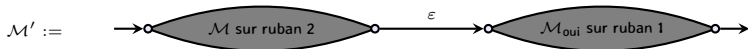


☞ On a donc une fonction  $\rho : [\mathcal{M}]_{code} \mapsto [\mathcal{M}']_{code}$  **calculable** telle que

$$\forall \mathcal{M}, \mathcal{M} \in \mathcal{H}_\epsilon \Leftrightarrow \mathcal{M}' \in \mathcal{P}.$$

☞ Cette fonction est donc une réduction  $\mathcal{H}_\epsilon \rightarrow \mathcal{P}$ .





☞ On a donc une fonction  $\rho : [\mathcal{M}]_{code} \mapsto [\mathcal{M}']_{code}$  **calculable** telle que

$$\forall \mathcal{M}, \mathcal{M} \in \mathcal{H}_\epsilon \Leftrightarrow \mathcal{M}' \in \mathcal{P}.$$

☞ Cette fonction est donc une réduction  $\mathcal{H}_\epsilon \rightarrow \mathcal{P}$ .

☞ Comme  $\mathcal{H}_\epsilon$  est indécidable,  $\mathcal{P}$  est également indécidable. □

# Fin de la preuve du théorème de Rice

On vient d'établir le lemme suivant :

lemme

Soit  $\mathcal{P} = \langle I, P \rangle$  un problème de langage des machines de Turing sur  $\Sigma$ , tel que  $\mathcal{P}$  est non-trivial et  $\mathcal{M}_\emptyset \notin P$ . Alors  $\mathcal{P}$  est indécidable.

# Fin de la preuve du théorème de Rice

On vient d'établir le lemme suivant :

lemme

Soit  $\mathcal{P} = \langle I, P \rangle$  un problème de langage des machines de Turing sur  $\Sigma$ , tel que  $\mathcal{P}$  est non-trivial et  $\mathcal{M}_\emptyset \notin P$ . Alors  $\mathcal{P}$  est indécidable.

Et si  $\mathcal{M}_\emptyset \in P$ ?

# Fin de la preuve du théorème de Rice

On vient d'établir le lemme suivant :

lemme

Soit  $\mathcal{P} = \langle I, P \rangle$  un problème de langage des machines de Turing sur  $\Sigma$ , tel que  $\mathcal{P}$  est non-trivial et  $\mathcal{M}_\emptyset \notin P$ . Alors  $\mathcal{P}$  est indécidable.

Et si  $\mathcal{M}_\emptyset \in P$ ?

☞ On note  $P' = I \setminus P$ , et on considère le problème  $\mathcal{P}' = \langle I, P' \rangle$ .

□

On vient d'établir le lemme suivant :

lemme

Soit  $\mathcal{P} = \langle I, P \rangle$  un problème de langage des machines de Turing sur  $\Sigma$ , tel que  $\mathcal{P}$  est non-trivial et  $\mathcal{M}_\emptyset \notin P$ . Alors  $\mathcal{P}$  est indécidable.

Et si  $\mathcal{M}_\emptyset \in P$ ?

☞ On note  $P' = I \setminus P$ , et on considère le problème  $\mathcal{P}' = \langle I, P' \rangle$ .

☞ Comme  $\mathcal{P}$  est non-trivial, il existe  $i_-, i_+ \in I$  tels que  $i_- \notin P$  et  $i_+ \in P$ .

□

On vient d'établir le lemme suivant :

lemme

Soit  $\mathcal{P} = \langle I, P \rangle$  un problème de langage des machines de Turing sur  $\Sigma$ , tel que  $\mathcal{P}$  est non-trivial et  $\mathcal{M}_\emptyset \notin P$ . Alors  $\mathcal{P}$  est indécidable.

Et si  $\mathcal{M}_\emptyset \in P$ ?

☞ On note  $P' = I \setminus P$ , et on considère le problème  $\mathcal{P}' = \langle I, P' \rangle$ .

☞ Comme  $\mathcal{P}$  est non-trivial, il existe  $i_-, i_+ \in I$  tels que  $i_- \notin P$  et  $i_+ \in P$ .

☞ Donc  $\mathcal{P}'$  est également non-trivial, puisque  $i_- \in I \setminus P = P'$  et  $i_+ \notin I \setminus P = P'$ .

□

On vient d'établir le lemme suivant :

lemme

Soit  $\mathcal{P} = \langle I, P \rangle$  un problème de langage des machines de Turing sur  $\Sigma$ , tel que  $\mathcal{P}$  est non-trivial et  $\mathcal{M}_\emptyset \notin P$ . Alors  $\mathcal{P}$  est indécidable.

Et si  $\mathcal{M}_\emptyset \in P$ ?

- ☞ On note  $P' = I \setminus P$ , et on considère le problème  $\mathcal{P}' = \langle I, P' \rangle$ .
- ☞ Comme  $\mathcal{P}$  est non-trivial, il existe  $i_-, i_+ \in I$  tels que  $i_- \notin P$  et  $i_+ \in P$ .
- ☞ Donc  $\mathcal{P}'$  est également non-trivial, puisque  $i_- \in I \setminus P = P'$  et  $i_+ \notin I \setminus P = P'$ .
- ☞ Soient deux machines  $\mathcal{M}_1, \mathcal{M}_2$  telles que  $\mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2)$  :

$$\mathcal{M}_1 \in P'$$

□

On vient d'établir le lemme suivant :

lemme

Soit  $\mathcal{P} = \langle I, P \rangle$  un problème de langage des machines de Turing sur  $\Sigma$ , tel que  $\mathcal{P}$  est non-trivial et  $\mathcal{M}_\emptyset \notin P$ . Alors  $\mathcal{P}$  est indécidable.

Et si  $\mathcal{M}_\emptyset \in P$ ?

- ☞ On note  $P' = I \setminus P$ , et on considère le problème  $\mathcal{P}' = \langle I, P' \rangle$ .
- ☞ Comme  $\mathcal{P}$  est non-trivial, il existe  $i_-, i_+ \in I$  tels que  $i_- \notin P$  et  $i_+ \in P$ .
- ☞ Donc  $\mathcal{P}'$  est également non-trivial, puisque  $i_- \in I \setminus P = P'$  et  $i_+ \notin I \setminus P = P'$ .
- ☞ Soient deux machines  $\mathcal{M}_1, \mathcal{M}_2$  telles que  $\mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2)$  :

$$\mathcal{M}_1 \in P' \Leftrightarrow \mathcal{M}_1 \notin P$$

□



On vient d'établir le lemme suivant :

lemme

Soit  $\mathcal{P} = \langle I, P \rangle$  un problème de langage des machines de Turing sur  $\Sigma$ , tel que  $\mathcal{P}$  est non-trivial et  $\mathcal{M}_\emptyset \notin P$ . Alors  $\mathcal{P}$  est indécidable.

Et si  $\mathcal{M}_\emptyset \in P$ ?

- ☞ On note  $P' = I \setminus P$ , et on considère le problème  $\mathcal{P}' = \langle I, P' \rangle$ .
- ☞ Comme  $\mathcal{P}$  est non-trivial, il existe  $i_-, i_+ \in I$  tels que  $i_- \notin P$  et  $i_+ \in P$ .
- ☞ Donc  $\mathcal{P}'$  est également non-trivial, puisque  $i_- \in I \setminus P = P'$  et  $i_+ \notin I \setminus P = P'$ .
- ☞ Soient deux machines  $\mathcal{M}_1, \mathcal{M}_2$  telles que  $\mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2)$  :

$$\mathcal{M}_1 \in P' \Leftrightarrow \mathcal{M}_1 \notin P \Leftrightarrow \mathcal{M}_2 \notin P$$

□

On vient d'établir le lemme suivant :

lemme

Soit  $\mathcal{P} = \langle I, P \rangle$  un problème de langage des machines de Turing sur  $\Sigma$ , tel que  $\mathcal{P}$  est non-trivial et  $\mathcal{M}_\emptyset \notin P$ . Alors  $\mathcal{P}$  est indécidable.

Et si  $\mathcal{M}_\emptyset \in P$ ?

- ☞ On note  $P' = I \setminus P$ , et on considère le problème  $\mathcal{P}' = \langle I, P' \rangle$ .
- ☞ Comme  $\mathcal{P}$  est non-trivial, il existe  $i_-, i_+ \in I$  tels que  $i_- \notin P$  et  $i_+ \in P$ .
- ☞ Donc  $\mathcal{P}'$  est également non-trivial, puisque  $i_- \in I \setminus P = P'$  et  $i_+ \notin I \setminus P = P'$ .
- ☞ Soient deux machines  $\mathcal{M}_1, \mathcal{M}_2$  telles que  $\mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2)$  :

$$\mathcal{M}_1 \in P' \Leftrightarrow \mathcal{M}_1 \notin P \Leftrightarrow \mathcal{M}_2 \notin P \Leftrightarrow \mathcal{M}_2 \in P'$$

□

On vient d'établir le lemme suivant :

lemme

Soit  $\mathcal{P} = \langle I, P \rangle$  un problème de langage des machines de Turing sur  $\Sigma$ , tel que  $\mathcal{P}$  est non-trivial et  $\mathcal{M}_\emptyset \notin P$ . Alors  $\mathcal{P}$  est indécidable.

Et si  $\mathcal{M}_\emptyset \in P$ ?

☞ On note  $P' = I \setminus P$ , et on considère le problème  $\mathcal{P}' = \langle I, P' \rangle$ .

☞ Comme  $\mathcal{P}$  est non-trivial, il existe  $i_-, i_+ \in I$  tels que  $i_- \notin P$  et  $i_+ \in P$ .

☞ Donc  $\mathcal{P}'$  est également non-trivial, puisque  $i_- \in I \setminus P = P'$  et  $i_+ \notin I \setminus P = P'$ .

☞ Soient deux machines  $\mathcal{M}_1, \mathcal{M}_2$  telles que  $\mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2)$  :

$$\mathcal{M}_1 \in P' \Leftrightarrow \mathcal{M}_1 \notin P \Leftrightarrow \mathcal{M}_2 \notin P \Leftrightarrow \mathcal{M}_2 \in P'$$

☞ Donc  $\mathcal{P}'$  est un problème de langages.

□

On vient d'établir le lemme suivant :

lemme

Soit  $\mathcal{P} = \langle I, P \rangle$  un problème de langage des machines de Turing sur  $\Sigma$ , tel que  $\mathcal{P}$  est non-trivial et  $\mathcal{M}_\emptyset \notin P$ . Alors  $\mathcal{P}$  est indécidable.

Et si  $\mathcal{M}_\emptyset \in P$ ?

☞ On note  $P' = I \setminus P$ , et on considère le problème  $\mathcal{P}' = \langle I, P' \rangle$ .

☞ Comme  $\mathcal{P}$  est non-trivial, il existe  $i_-, i_+ \in I$  tels que  $i_- \notin P$  et  $i_+ \in P$ .

☞ Donc  $\mathcal{P}'$  est également non-trivial, puisque  $i_- \in I \setminus P = P'$  et  $i_+ \notin I \setminus P = P'$ .

☞ Soient deux machines  $\mathcal{M}_1, \mathcal{M}_2$  telles que  $\mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2)$  :

$$\mathcal{M}_1 \in P' \Leftrightarrow \mathcal{M}_1 \notin P \Leftrightarrow \mathcal{M}_2 \notin P \Leftrightarrow \mathcal{M}_2 \in P'$$

☞ Donc  $\mathcal{P}'$  est un problème de langages.

☞ Puisque  $\mathcal{M}_\emptyset \in P$ , on sait que  $\mathcal{M}_\emptyset \notin P'$ .

□

On vient d'établir le lemme suivant :

lemme

Soit  $\mathcal{P} = \langle I, P \rangle$  un problème de langage des machines de Turing sur  $\Sigma$ , tel que  $\mathcal{P}$  est non-trivial et  $\mathcal{M}_\emptyset \notin P$ . Alors  $\mathcal{P}$  est indécidable.

Et si  $\mathcal{M}_\emptyset \in P$ ?

- ☞ On note  $P' = I \setminus P$ , et on considère le problème  $\mathcal{P}' = \langle I, P' \rangle$ .
- ☞ Comme  $\mathcal{P}$  est non-trivial, il existe  $i_-, i_+ \in I$  tels que  $i_- \notin P$  et  $i_+ \in P$ .
- ☞ Donc  $\mathcal{P}'$  est également non-trivial, puisque  $i_- \in I \setminus P = P'$  et  $i_+ \notin I \setminus P = P'$ .
- ☞ Soient deux machines  $\mathcal{M}_1, \mathcal{M}_2$  telles que  $\mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2)$  :

$$\mathcal{M}_1 \in P' \Leftrightarrow \mathcal{M}_1 \notin P \Leftrightarrow \mathcal{M}_2 \notin P \Leftrightarrow \mathcal{M}_2 \in P'$$

- ☞ Donc  $\mathcal{P}'$  est un problème de langages.
- ☞ Puisque  $\mathcal{M}_\emptyset \in P$ , on sait que  $\mathcal{M}_\emptyset \notin P'$ .
- ☞ On peut donc appliquer le lemme, et obtenir que  $\mathcal{P}'$  est indécidable.

□

On vient d'établir le lemme suivant :

lemme

Soit  $\mathcal{P} = \langle I, P \rangle$  un problème de langage des machines de Turing sur  $\Sigma$ , tel que  $\mathcal{P}$  est non-trivial et  $\mathcal{M}_\emptyset \notin P$ . Alors  $\mathcal{P}$  est indécidable.

Et si  $\mathcal{M}_\emptyset \in P$ ?

- ☞ On note  $P' = I \setminus P$ , et on considère le problème  $\mathcal{P}' = \langle I, P' \rangle$ .
- ☞ Comme  $\mathcal{P}$  est non-trivial, il existe  $i_-, i_+ \in I$  tels que  $i_- \notin P$  et  $i_+ \in P$ .
- ☞ Donc  $\mathcal{P}'$  est également non-trivial, puisque  $i_- \in I \setminus P = P'$  et  $i_+ \notin I \setminus P = P'$ .
- ☞ Soient deux machines  $\mathcal{M}_1, \mathcal{M}_2$  telles que  $\mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2)$  :

$$\mathcal{M}_1 \in P' \Leftrightarrow \mathcal{M}_1 \notin P \Leftrightarrow \mathcal{M}_2 \notin P \Leftrightarrow \mathcal{M}_2 \in P'$$

- ☞ Donc  $\mathcal{P}'$  est un problème de langages.
- ☞ Puisque  $\mathcal{M}_\emptyset \in P$ , on sait que  $\mathcal{M}_\emptyset \notin P'$ .
- ☞ On peut donc appliquer le lemme, et obtenir que  $\mathcal{P}'$  est indécidable.
- ☞ Le complémentaire d'un problème indécidable étant indécidable,  $\mathcal{P}$  est indécidable.

□