UPEC
UNIVERSITÉ PARIS-EST CRÉTEIL
EPISEN

# Introduction

## Calculabilité et Complexité

Paul Brunet

# Problématique

Quelles questions sont solubles à l'aide d'un dispositif automatique ?

# Problématique

Quelles questions sont solubles à l'aide d'un dispositif automatique ?

Quelles questions ne sont pas solubles à l'aide d'un dispositif automatique ?

# Problématique

Quelles questions sont solubles à l'aide d'un dispositif automatique ?

Quelles questions ne sont pas solubles à l'aide d'un dispositif automatique ?

Étant donné un problème $P$ :

# Problématique

Quelles questions sont solubles à l'aide d'un dispositif automatique ?

Quelles questions ne sont pas solubles à l'aide d'un dispositif automatique ?

Étant donné un problème $P$ :

☞ soit il existe un programme qui résout $P$ : il « suffit » d'écrire ce programme ;

# Problématique

Quelles questions sont solubles à l'aide d'un dispositif automatique ?

Quelles questions ne sont pas solubles à l'aide d'un dispositif automatique ?

Étant donné un problème $P$ :

- soit il existe un programme qui résout $P$ : il « suffit » d'écrire ce programme ;
- soit il n'en existe pas : ???

# Problématique

Quelles questions sont solubles à l'aide d'un dispositif automatique ?

Quelles questions ne sont pas solubles à l'aide d'un dispositif automatique ?

Étant donné un problème $P$ :

- ☞ soit il existe un programme qui résout $P$ : il « suffit » d'écrire ce programme ;
- ☞ soit il n'en existe pas : ???

☞ De quel genre de questions parle-t'on ?

# Problématique

Quelles questions sont solubles à l'aide d'un dispositif automatique ?

Quelles questions ne sont pas solubles à l'aide d'un dispositif automatique ?

Étant donné un problème $P$ :

- ☞ soit il existe un programme qui résout $P$ : il « suffit » d'écrire ce programme ;
- ☞ soit il n'en existe pas : ???

- ☞ De quel genre de questions parle-t'on ?
- ☞ Quels genres de dispositifs s'autorise t'on ?

# Introduction
## Plan

1. Problèmes de décision

2. Modèles de calcul

3. Théorie des automates

# De quel genre de questions parle-t'on ?

Pour l'essentiel du cours : problèmes de décision.

**définition**

Un problème de décision est une paire $\mathcal{P} := \langle I, P \rangle$ consitituée

- ☞ d'un ensemble $I$ d'instances ;
- ☞ et d'un sous ensemble $E \subseteq I$ d'instances positives.

# De quel genre de questions parle-t'on ?

Pour l'essentiel du cours : problèmes de décision.

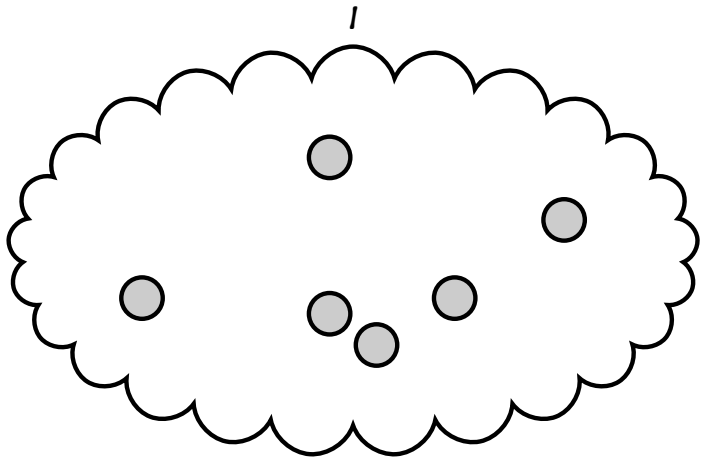
**définition**

Un problème de décision est une paire $\mathcal{P} := \langle I, P \rangle$ consitituée

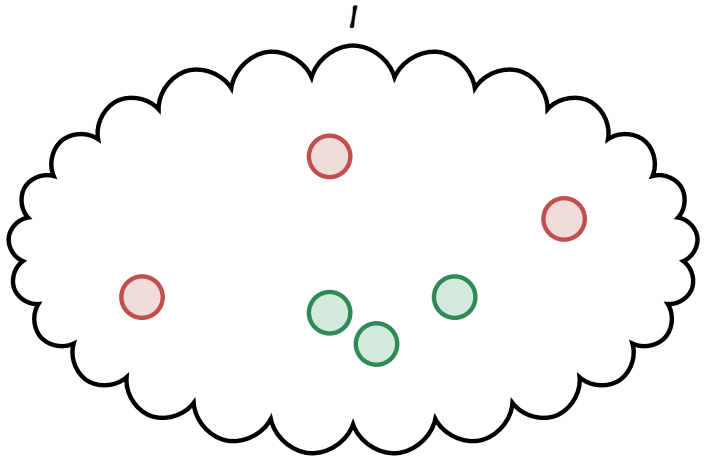
- ☞ d'un ensemble $I$ d'instances ;
- ☞ et d'un sous ensemble $E \subseteq I$ d'instances positives.

Résoudre un problème de décision signifie pouvoir déterminer en temps fini si $i \in P$, pour n'importe quelle instance $i \in I$.

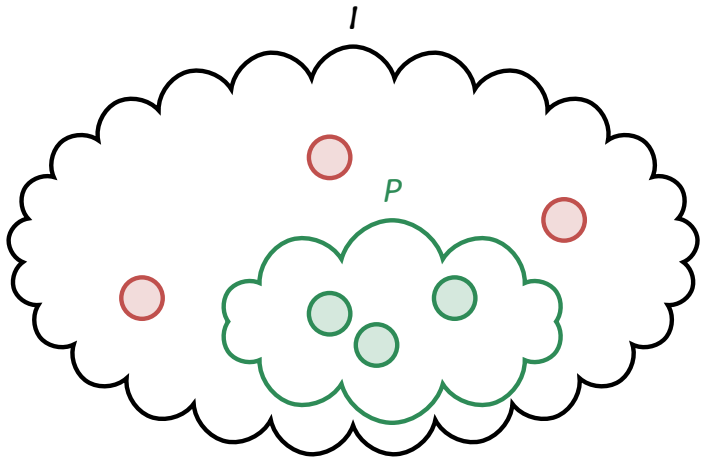
# Résoudre un problème de décision



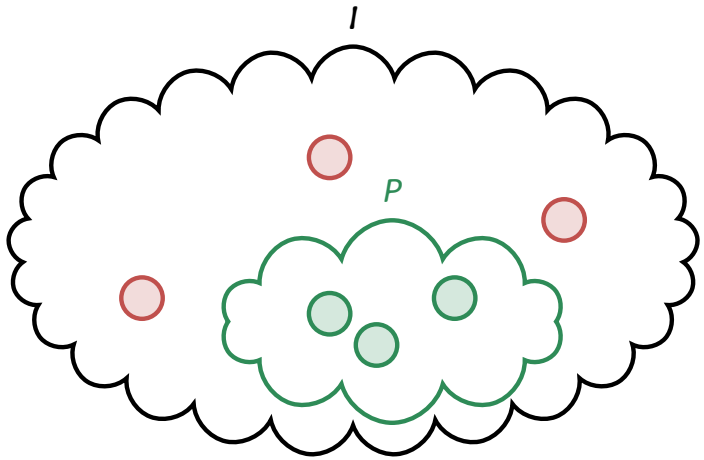
# Résoudre un problème de décision

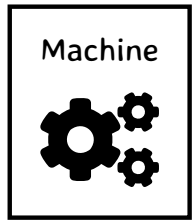


# Résoudre un problème de décision

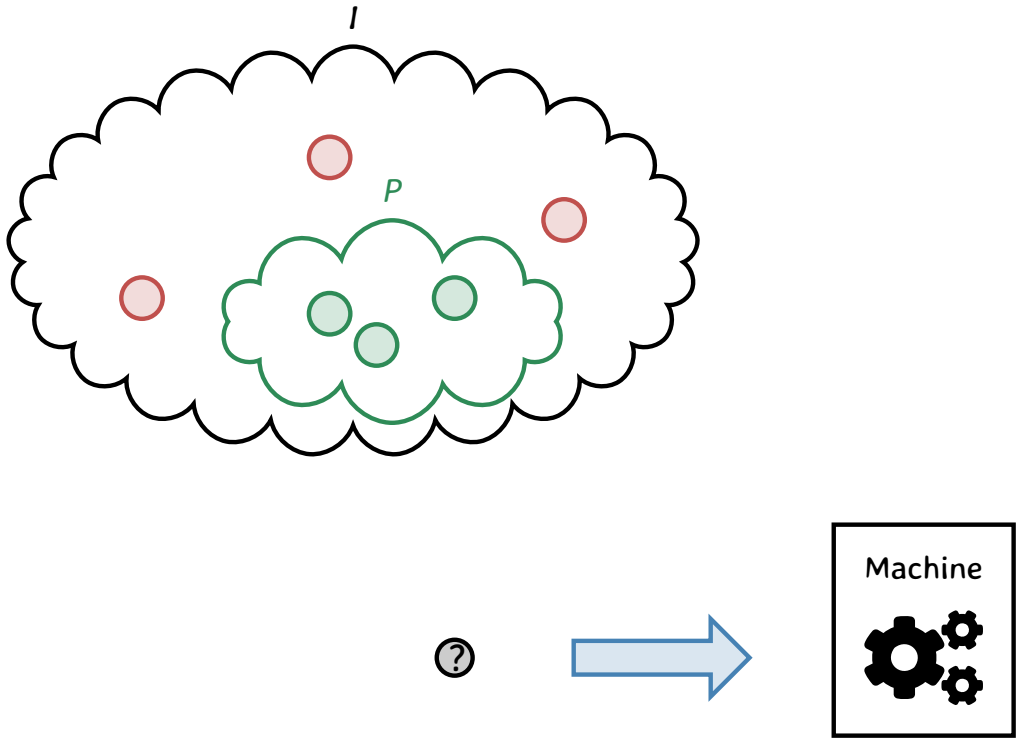


# Résoudre un problème de décision

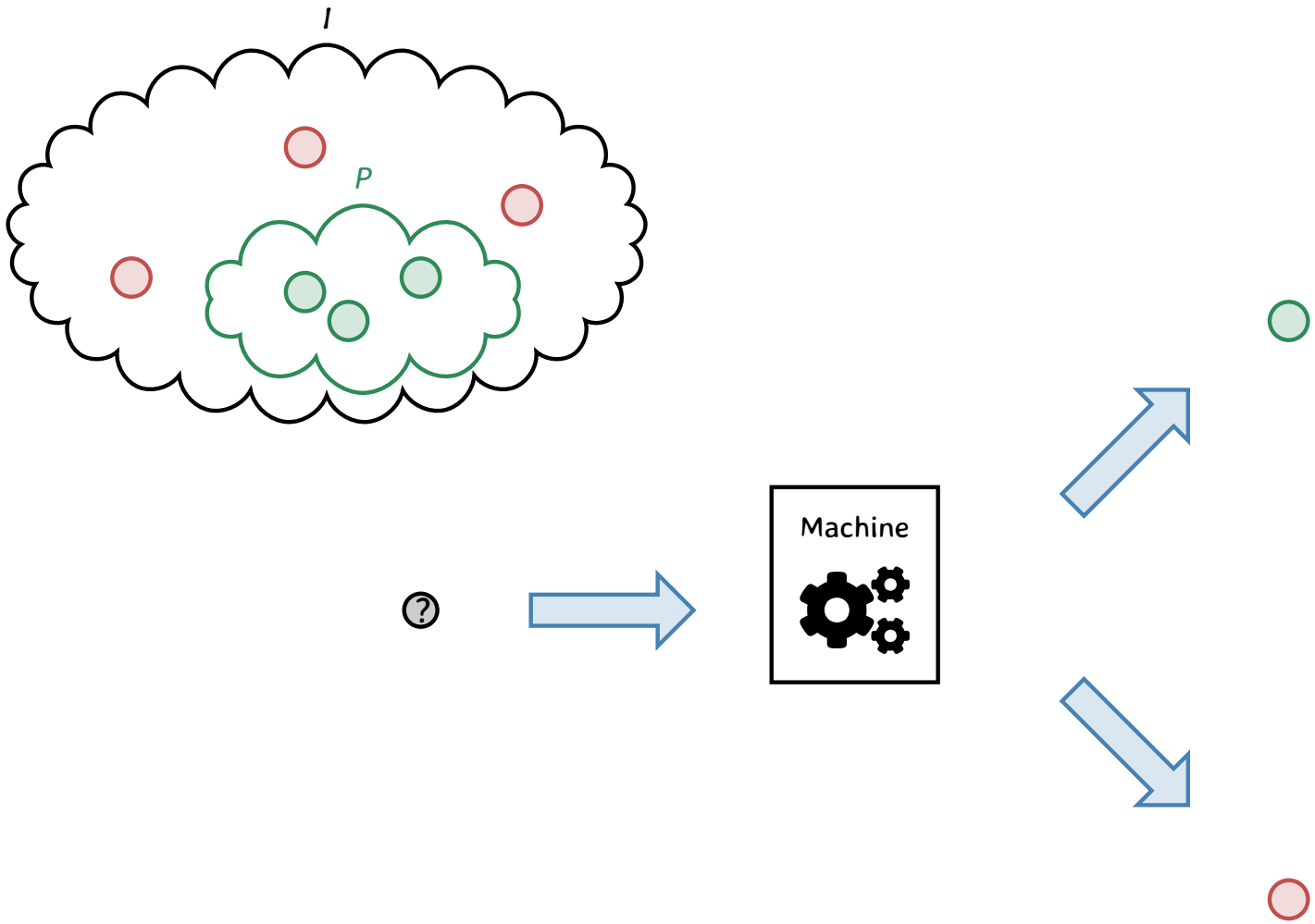


# Résoudre un problème de décision



# Résoudre un problème de décision



# PARITY
## exemple de problème de décision

$n$ **mod** 2 ?

**énoncé** Déterminer la parité d'un entier.

**question** soit $n \in \mathbb{N}$, $n$ est-il pair ?

# PARITY
## exemple de problème de décision

$$n \bmod 2 \,?$$

**énoncé** Déterminer la parité d'un entier.

**question** soit $n \in \mathbb{N}$, $n$ est-il pair ?

**instances** $I_{\text{PARITY}} := \mathbb{N}$.

**instances positives** $E_{\text{PARITY}} := \{2 \times k \mid k \in \mathbb{N}\}$.

définition (mots)

Un **alphabet** est un ensemble fini de symboles, souvent noté $\Sigma$.
Étant donné un tel alphabet $\Sigma$, l'ensemble des **mots** sur $\Sigma$, noté $\Sigma^\star$, est défini inductivement comme suit :

$$u, v \in \Sigma^\star ::= \varepsilon \mid a\,u$$

où $a \in \Sigma$. Autrement dit, un mot est soit le mot vide $\varepsilon$, soit un symbole $a$ suivi d'un mot $u$.

# Codages
## Comment utiliser une machine pour résoudre un problème

**définition (mots)**

Un **alphabet** est un ensemble fini de symboles, souvent noté $\Sigma$.
Étant donné un tel alphabet $\Sigma$, l'ensemble des **mots** sur $\Sigma$, noté $\Sigma^\star$, est défini inductivement comme suit :

$$u, v \in \Sigma^\star ::= \varepsilon \mid a\,u$$

où $a \in \Sigma$. Autrement dit, un mot est soit le mot vide $\varepsilon$, soit un symbole $a$ suivi d'un mot $u$. On peut **concaténer** deux mots $u$ et $v$ pour produire le mot $u \cdot v$ :

$$\varepsilon \cdot v := v \qquad\qquad (a\,u) \cdot v := a\,(u \cdot v)\,.$$

# Codages

## Comment utili

C'est la manière mathématique de parler de liste (une liste est soit la liste vide `[]`, soit un élément suivi d'une liste `a :: l`).

définition (mots)

Un **alphabet** est un ensemble fini de symboles, souvent noté $\Sigma$.
Étant donné un tel alphabet $\Sigma$, l'ensemble des **mots** sur $\Sigma$, noté $\Sigma^\star$, est défini inductivement comme suit :

$$u, v \in \Sigma^\star ::= \varepsilon \mid a\,u$$

où $a \in \Sigma$. Autrement dit, un mot est soit le mot vide $\varepsilon$, soit un symbole $a$ suivi d'un mot $u$. On peut **concaténer** deux mots $u$ et $v$ pour produire le mot $u \cdot v$ :

$$\varepsilon \cdot v := v \qquad\qquad (a\,u) \cdot v := a\,(u \cdot v)\,.$$

# Codages
## Comment utiliser une machine pour résoudre un problème

**définition (mots)**

Un **alphabet** est un ensemble fini de symboles, souvent noté $\Sigma$.
Étant donné un tel alphabet $\Sigma$, l'ensemble des **mots** sur $\Sigma$, noté $\Sigma^\star$, est défini inductivement comme suit :

$$u, v \in \Sigma^\star ::= \varepsilon \mid a\,u$$

où $a \in \Sigma$. Autrement dit, un mot est soit le mot vide $\varepsilon$, soit un symbole $a$ suivi d'un mot $u$. On peut **concaténer** deux mots $u$ et $v$ pour produire le mot $u \cdot v$ :

$$\varepsilon \cdot v := v \qquad\qquad (a\,u) \cdot v := a\,(u \cdot v)\,.$$

On a un problème de décision $\mathcal{P} = \langle I, P \rangle$.

**définition (codage)**

Un **codage** de $\mathcal{P}$ est donné par un alphabet fini $\Sigma$ et une fonction **injective**

$$\varphi : I \to \Sigma^\star.$$

# Codages
## Comment utiliser une machine pour résoudre un problème

**définition (mots)**

Un **alphabet** est un ensemble fini de symboles, souvent noté $\Sigma$.
Étant donné un tel alphabet $\Sigma$, l'ensemble des **mots** sur $\Sigma$, noté $\Sigma^\star$, est défini inductivement comme suit :

$$u, v \in \Sigma^\star ::= \varepsilon \mid a\,u$$

où $a \in \Sigma$. Autrement dit, un mot est soit le mot vide $\varepsilon$, soit un symbole $a$ suivi d'un mot $u$. On peut **concaténer** deux mots $u$ et $v$ pour produire le mot $u \cdot v$ :

$$\varepsilon \cdot v := v \qquad\qquad (a\,u) \cdot v := a\,(u \cdot v)\,.$$

On a un problème de décision $\mathcal{P} = \langle I, P \rangle$.

**définition (codage)**

Un **codage** de $\mathcal{P}$ est donné par un alphabet fini $\Sigma$ et une fonction **injective**

$$\varphi : I \to \Sigma^\star.$$

$$\forall i, j \in I,\ \varphi(i) = \varphi(j) \Rightarrow i = j.$$

# Codages
## Comment utiliser une machine pour résoudre un problème

définition (mots)

Un **alphabet** est un ensemble fini de symboles, souvent noté $\Sigma$.
Étant donné un tel alphabet $\Sigma$, l'ensemble des **mots** sur $\Sigma$, noté $\Sigma^\star$, est défini inductivement comme suit :

$$u, v \in \Sigma^\star ::= \varepsilon \mid a\,u$$

où $a \in \Sigma$. Autrement dit, un mot est soit le mot vide $\varepsilon$, soit un symbole $a$ suivi d'un mot $u$. On peut **concaténer** deux mots $u$ et $v$ pour produire le mot $u \cdot v$ :

$$\varepsilon \cdot v := v \qquad\qquad (a\,u) \cdot v := a\ (u \cdot v)\,.$$

On a un problème de décision $\mathcal{P} = \langle I, P \rangle$.

définition (codage)

Un **codage** de $\mathcal{P}$ est donné par un alphabet fini $\Sigma$ et une fonction **injective**

$$\forall i, j \in I,\ \varphi(i) = \varphi(j) \Rightarrow i = j.$$

$$\varphi : I \to \Sigma^\star.$$

Décider le problème $\mathcal{P}$ revient à reconnaitre le langage $L_{\mathcal{P}} := \{\varphi(i) \mid i \in P\}$.

# PARITY
## exemple de problème de décision

$$n \bmod 2 \,?$$

**énoncé** Déterminer la parité d'un entier.

**question** soit $n \in \mathbb{N}$, $n$ est-il pair ?

**instances** $I_{\text{PARITY}} := \mathbb{N}$.

**instances positives** $E_{\text{PARITY}} := \{2 \times k \mid k \in \mathbb{N}\}$.

**codage** $\lfloor - \rfloor_2 : \mathbb{N} \to \{0,1\}^\star$

**langage** $L_{\text{PARITY}} := \{\lfloor 2 \times k \rfloor_2 \mid k \in \mathbb{N}\}$

# PARITY
## exemple de problème de décision

$$n \bmod 2 \,?$$

**énoncé** Déterminer la parité d'un entier.

**question** soit $n \in \mathbb{N}$, $n$ est-il pair ?

**instances** $I_{\text{PARITY}} := \mathbb{N}$.

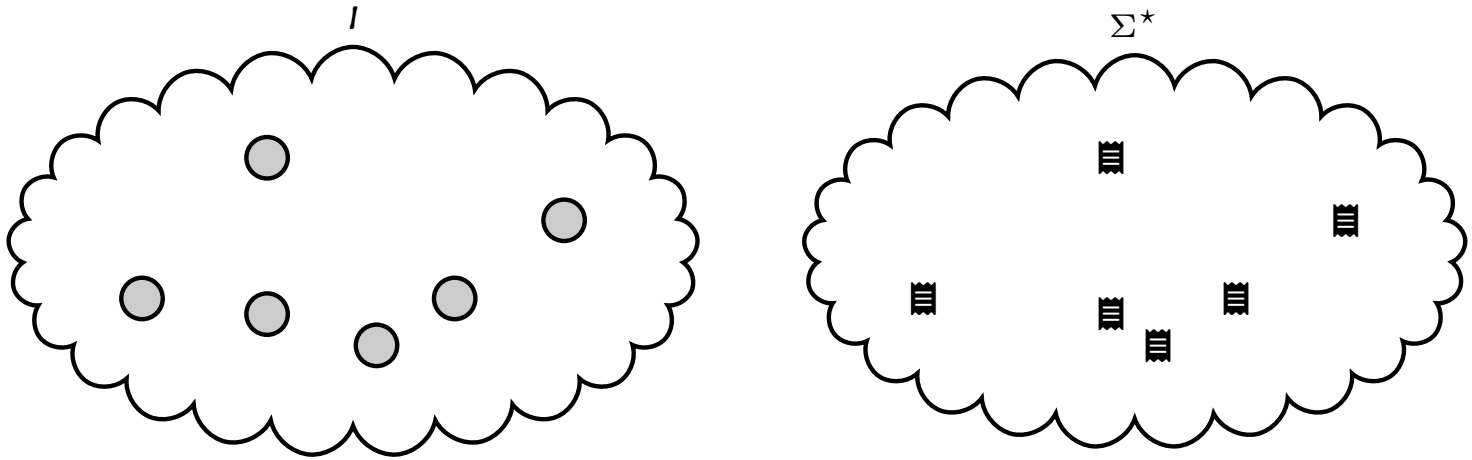
**instances positives** $E_{\text{PARITY}} := \{2 \times k \mid k \in \mathbb{N}\}$.

**codage** $\lfloor - \rfloor_2 : \mathbb{N} \to \{0,1\}^\star$

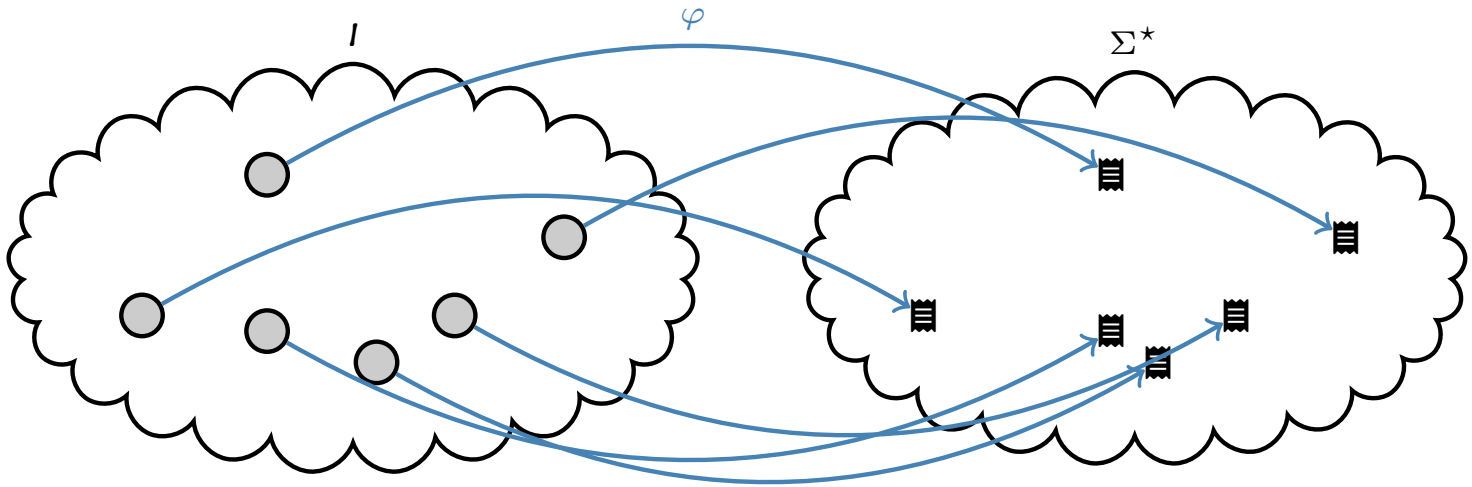
**langage** $L_{\text{PARITY}} := \{\lfloor 2 \times k \rfloor_2 \mid k \in \mathbb{N}\}$

Ici on remarque que $L_{\text{PARITY}} = \{w \in \{0,1\}^\star \mid w \text{ finit par un } 0\}$.

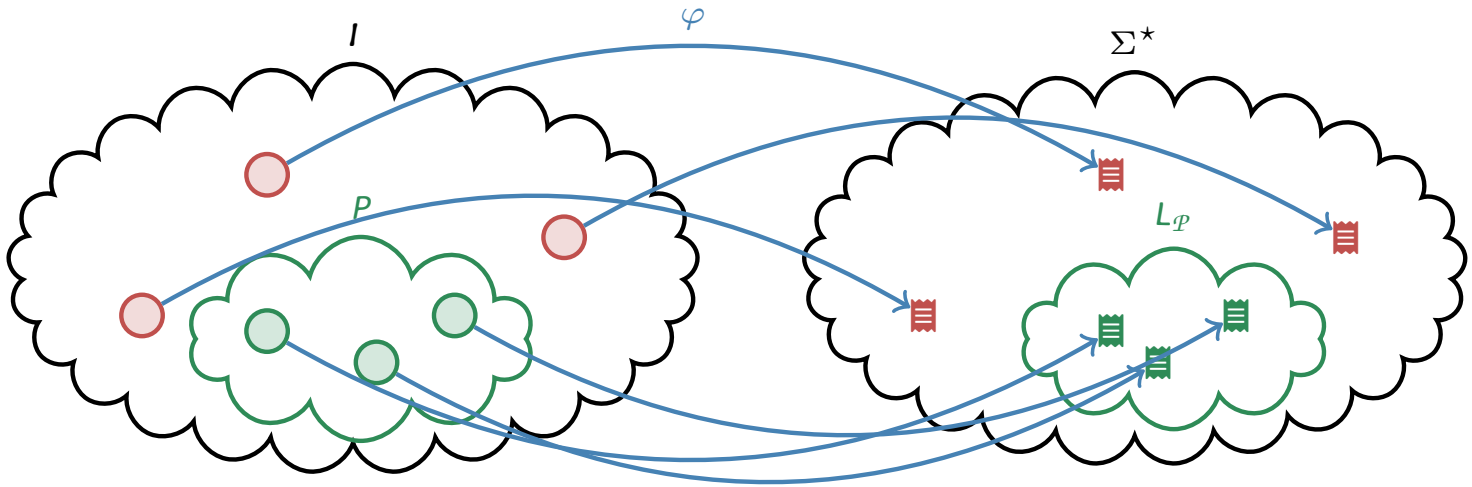
# Résoudre un problème de décision



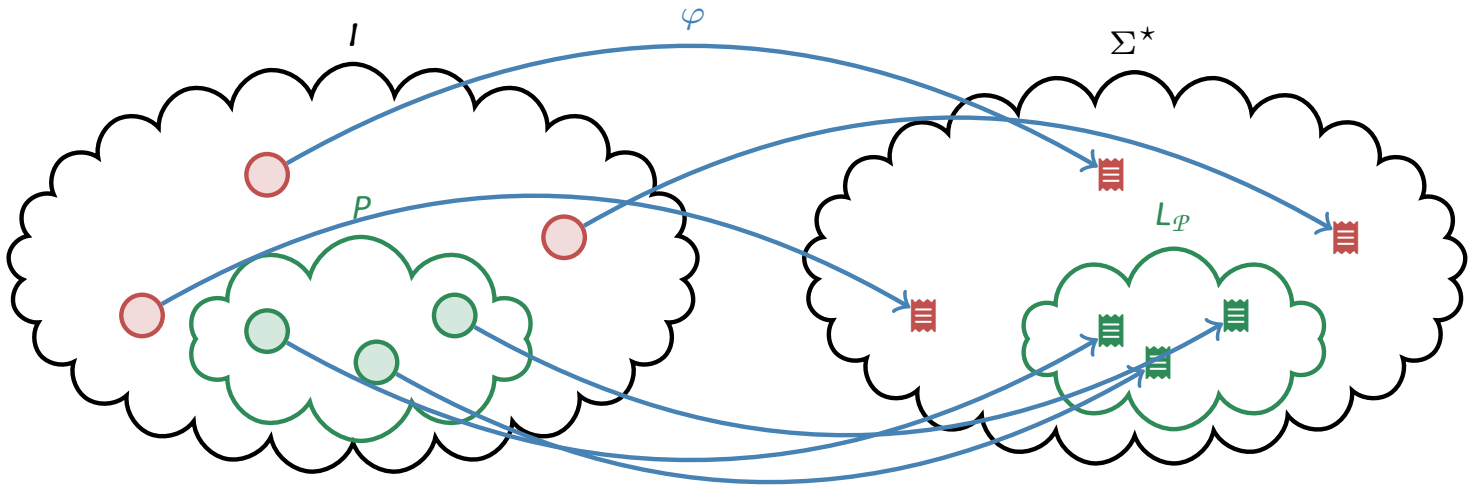
# Résoudre un problème de décision



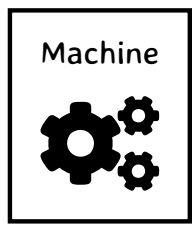
# Résoudre un problème de décision



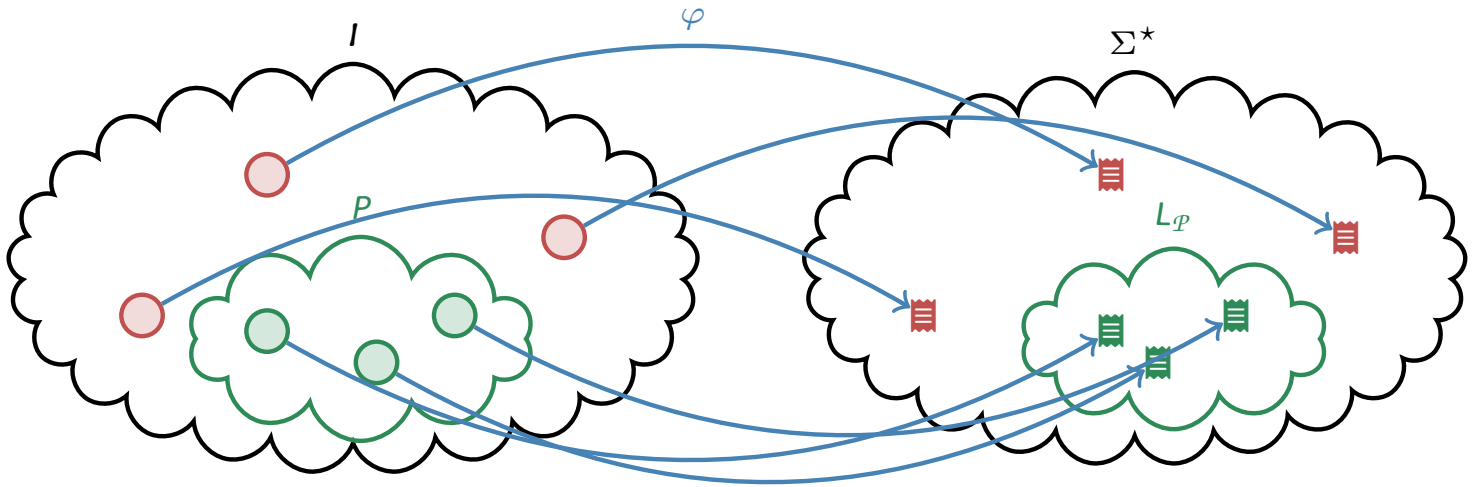
# Résoudre un problème de décision



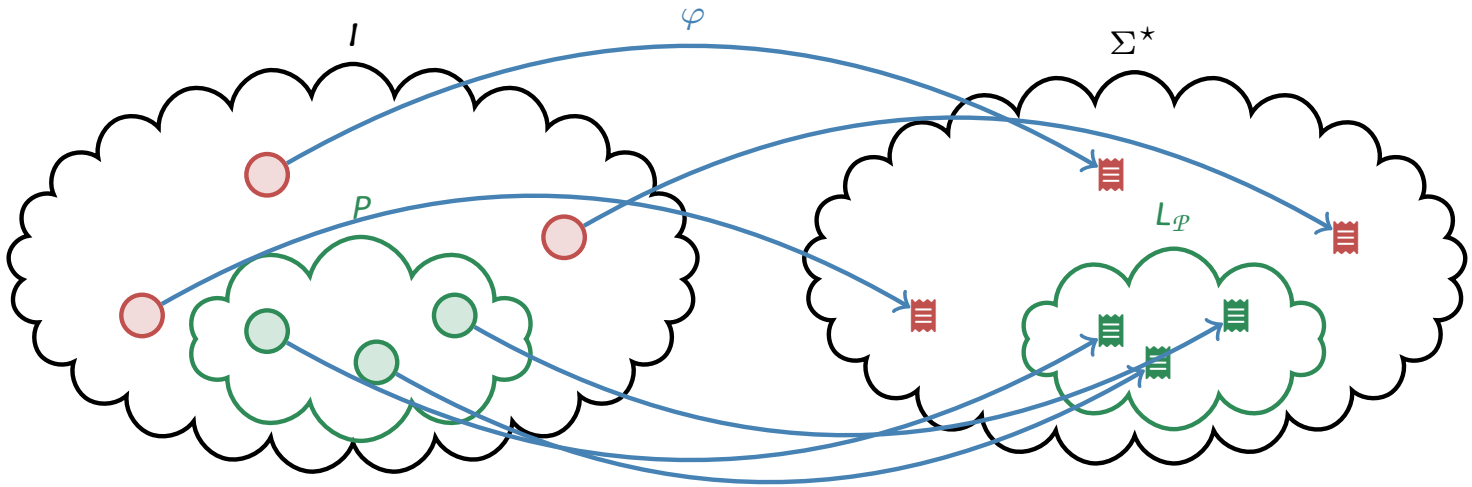

?

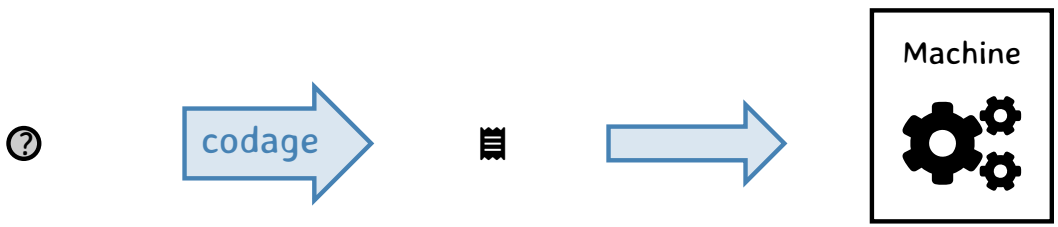


# Résoudre un problème de décision

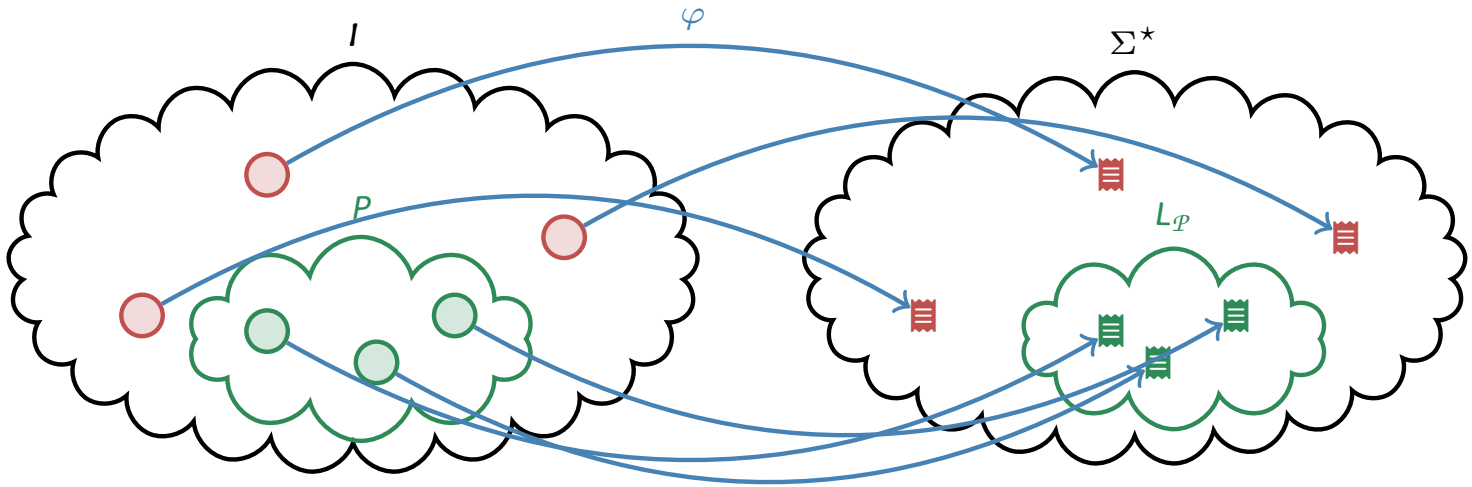


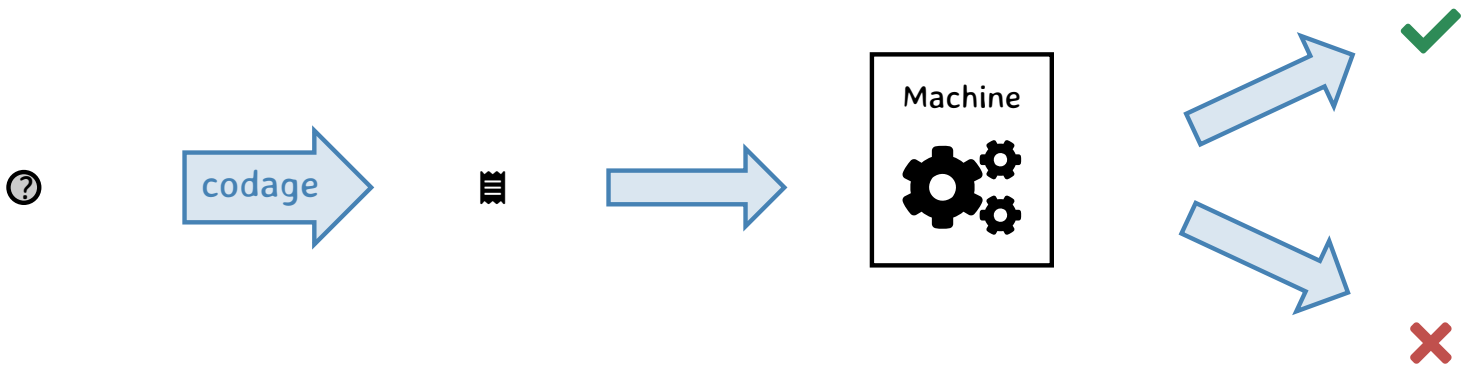

# Résoudre un problème de décision



# Résoudre un problème de décision



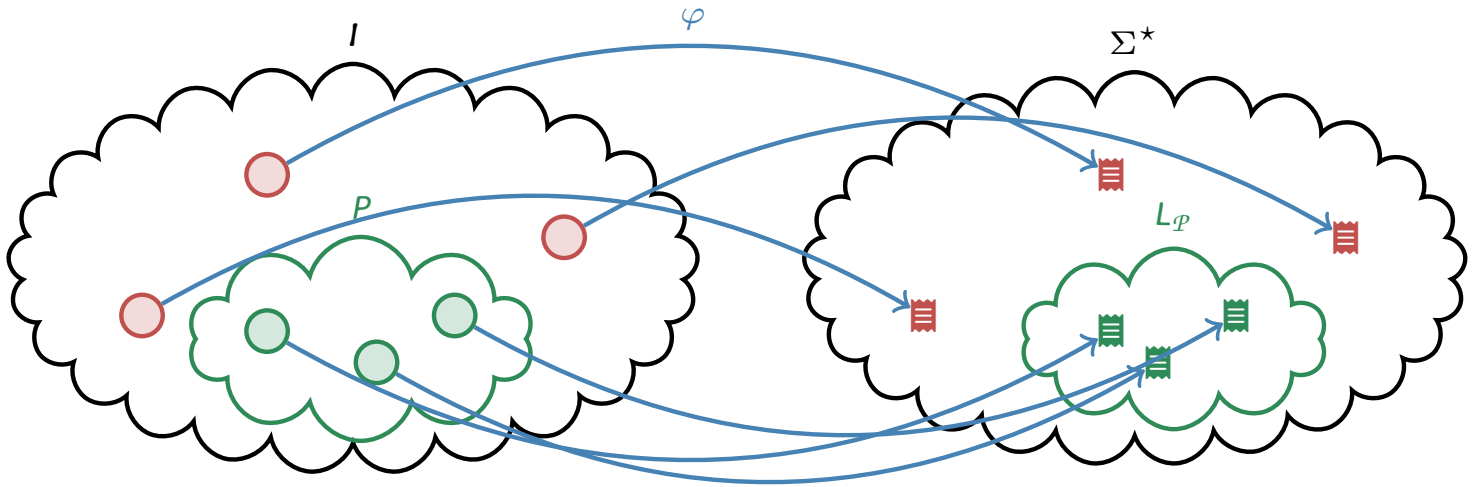
# Introduction
## Plan

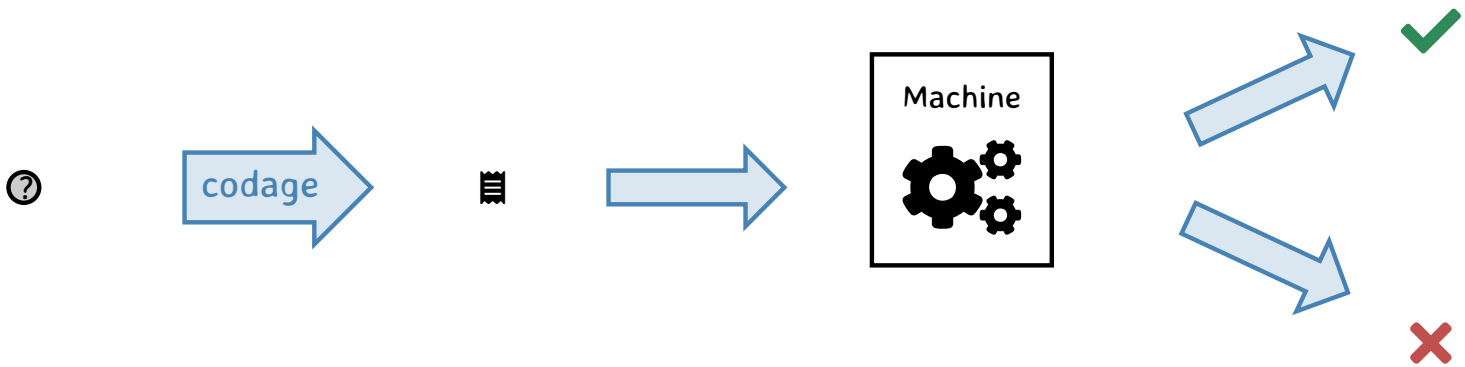
1. Problèmes de décision

2. Modèles de calcul

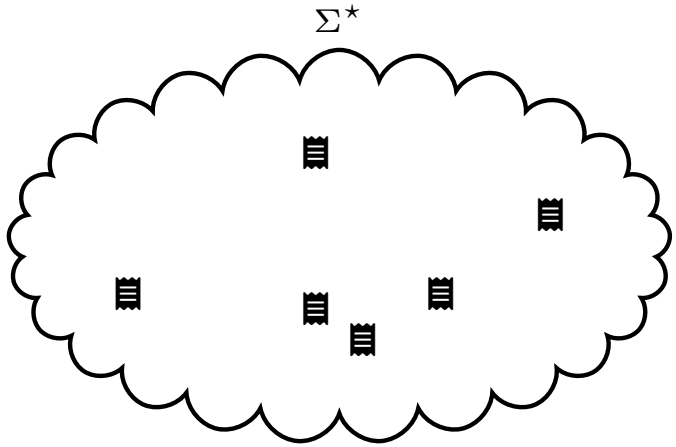
3. Théorie des automates

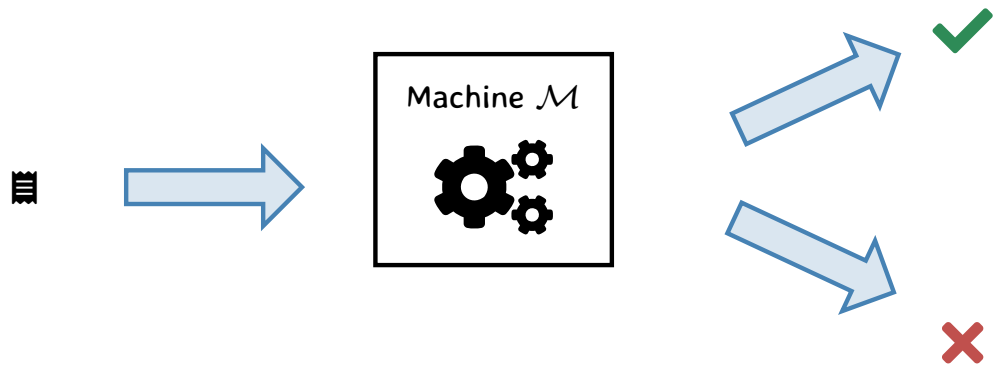
# Résoudre un problème de décision



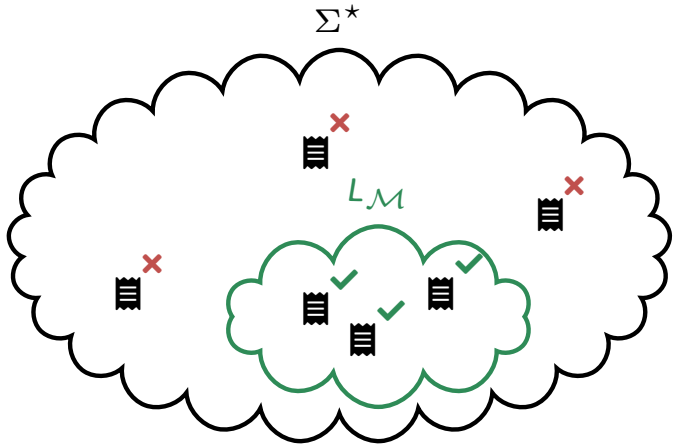



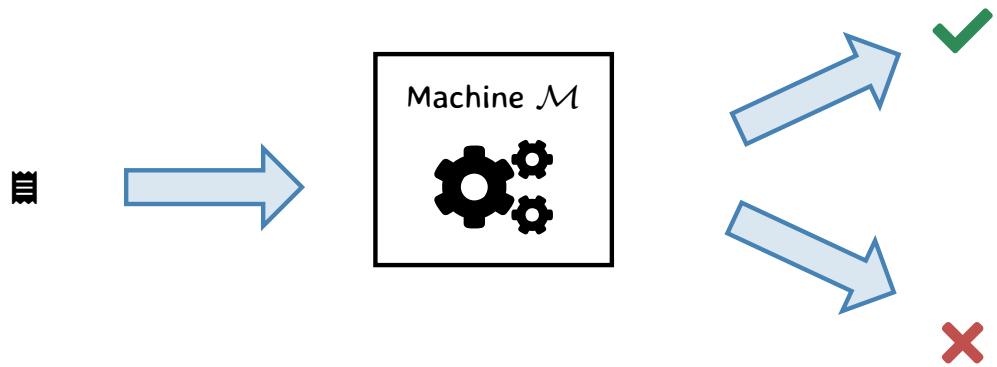
# Modèle de calcul



# Modèle de calcul



# Expressivité d'un modèle de calcul

définition (Modèle de calcul)

Un **modèle de calcul** sur l'alphabet $\Sigma$ est une paire $\mathscr{M} = \langle \mathbb{M}, \models \rangle$ où $\mathbb{M}$ est un ensemble de **machines**, et $\models \subseteq \Sigma^\star \times \mathbb{M}$ est une relation binaire d'**acceptation**.

# Expressivité d'un modèle de calcul

définition (Modèle de calcul)

Un **modèle de calcul** sur l'alphabet $\Sigma$ est une paire $\mathscr{M} = \langle \mathbb{M}, \models \rangle$ où $\mathbb{M}$ est un ensemble de **machines**, et $\models \subseteq \Sigma^\star \times \mathbb{M}$ est une relation binaire d'**acceptation**.

Une **machine** définit un **langage** :

$$L_{\mathcal{M}} := \{w \in \Sigma^\star \mid w \models \mathcal{M}\}\,.$$

# Expressivité d'un modèle de calcul

définition (Modèle de calcul)

Un **modèle de calcul** sur l'alphabet $\Sigma$ est une paire $\mathscr{M} = \langle \mathbb{M}, \models \rangle$ où $\mathbb{M}$ est un ensemble de **machines**, et $\models \subseteq \Sigma^\star \times \mathbb{M}$ est une relation binaire d'**acceptation**.

Une **machine** définit un **langage** :

$$L_{\mathcal{M}} := \{w \in \Sigma^\star \mid w \models \mathcal{M}\}\,.$$

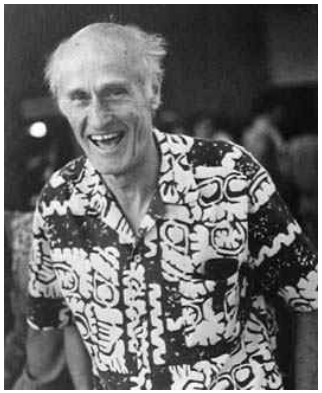
Un **modèle** définit une **classe de langages** :

$$\mathsf{Lang}\,(\mathscr{M}) := \{L_{\mathcal{M}} \subseteq \Sigma^\star \mid \mathcal{M} \in \mathbb{M}\}\,.$$

# Classes de langages

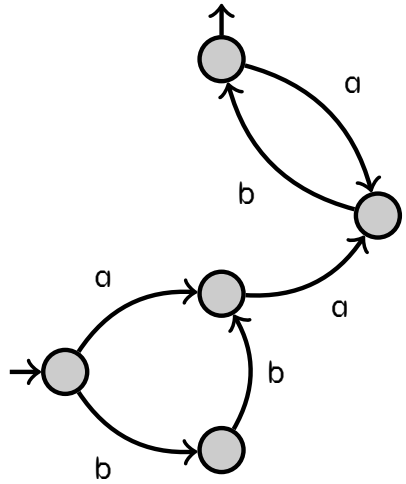
a, b, aa, ab, ba, bb

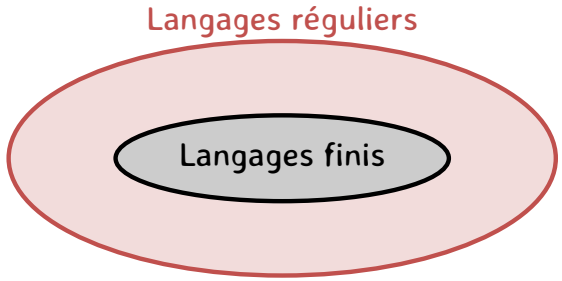
# Classes de langages



Stephen Cole Kleene

## Automates



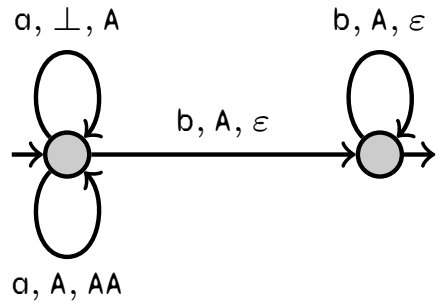



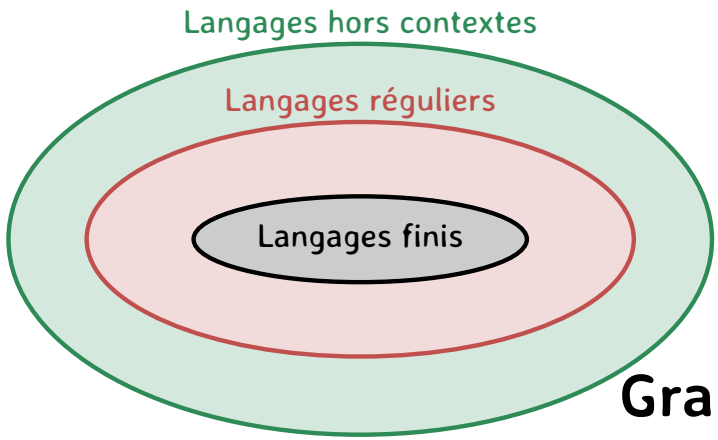

## Expressions régulières

$$(a|bb)\ ab\ (ab)^\star$$

# Classes de langages

## Automates à piles



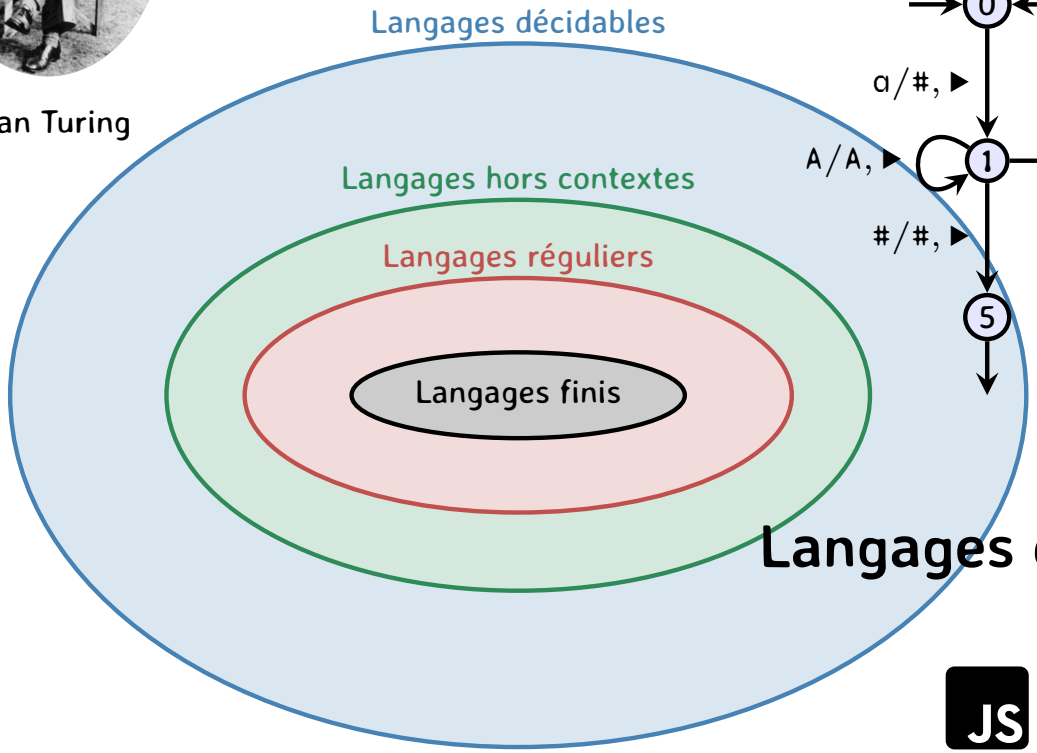




## Grammaires Hors-Contexte

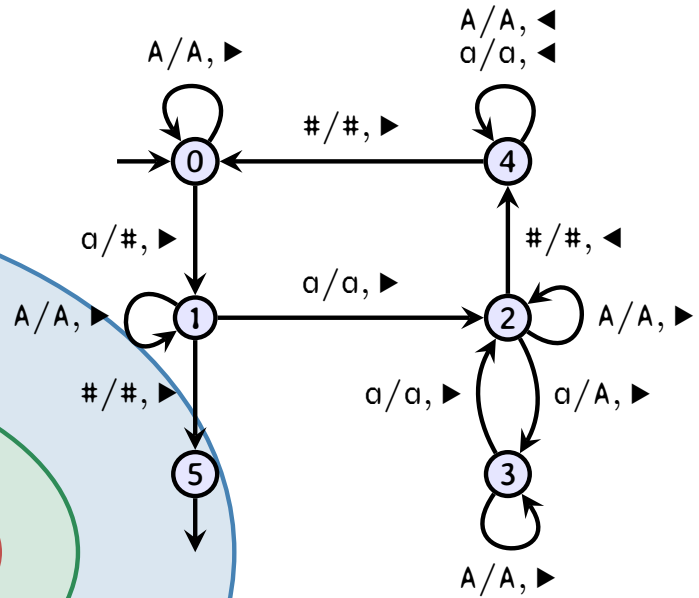
$S \to \varepsilon \mid aSb$

# Classes de langages

Alan Turing




## Machines de Turing




## Langages de programmation




# Comparaison de modèles

**définition (Comparaison de modèles)**

Soient $\mathscr{M}_1$ et $\mathscr{M}_2$ deux modèles de calcul.

☞ $\mathscr{M}_1$ se réduit à $\mathscr{M}_2$, noté $\mathscr{M}_1 \preceq \mathscr{M}_2$, si pour toute machine $\mathcal{M} \in \mathbb{M}_1$ il existe une machine $\mathcal{M}' \in \mathbb{M}_2$ acceptant le même langage, autrement dit :

$$\forall \mathcal{M} \in \mathbb{M}_1,\ \exists \mathcal{M}' \in \mathbb{M}_2 :\ \forall w \in \Sigma^\star,\ w \models_1 \mathcal{M} \Leftrightarrow w \models_2 \mathcal{M}'.$$

☞ $\mathscr{M}_1$ est équivalent à $\mathscr{M}_2$, noté $\mathscr{M}_1 \approx \mathscr{M}_2$, si $\mathscr{M}_1 \preceq \mathscr{M}_2$ et $\mathscr{M}_2 \preceq \mathscr{M}_1$.

# Comparaison de modèles

**définition (Comparaison de modèles)**

Soient $\mathscr{M}_1$ et $\mathscr{M}_2$ deux modèles de calcul.

☞ $\mathscr{M}_1$ se réduit à $\mathscr{M}_2$, noté $\mathscr{M}_1 \preceq \mathscr{M}_2$, si pour toute machine $\mathcal{M} \in \mathbb{M}_1$ il existe une machine $\mathcal{M}' \in \mathbb{M}_2$ acceptant le même langage, autrement dit :

$$\forall \mathcal{M} \in \mathbb{M}_1,\ \exists \mathcal{M}' \in \mathbb{M}_2 :\ \forall w \in \Sigma^\star,\ w \models_1 \mathcal{M} \Leftrightarrow w \models_2 \mathcal{M}'.$$

☞ $\mathscr{M}_1$ est équivalent à $\mathscr{M}_2$, noté $\mathscr{M}_1 \approx \mathscr{M}_2$, si $\mathscr{M}_1 \preceq \mathscr{M}_2$ et $\mathscr{M}_2 \preceq \mathscr{M}_1$.

**remarque**

Pour toute paire de modèles $\mathscr{M}_1$, $\mathscr{M}_2$ on a :

$$\mathscr{M}_1 \preceq \mathscr{M}_2 \Leftrightarrow \text{Lang}\,(\mathscr{M}_1) \subseteq \text{Lang}\,(\mathscr{M}_2)$$

# Hierarchies

- ☞ Hierarchies de problèmes
- ☞ Hierarchies de langages
- ☞ Hierarchies de modèles

Un problème **difficile** correspond à un langage **difficile**, et est soluble par un modèle **expressif/puissant**.
Un problème **facile** correspond à un langage **facile**, et est soluble par un modèle **simple**.

# Hierarchies

- ☞ Hierarchies de problèmes
- ☞ Hierarchies de langages
- ☞ Hierarchies de modèles

Un problème **difficile** correspond à un langage **difficile**, et est soluble par un modèle **expressif/puissant**.
Un problème **facile** correspond à un langage **facile**, et est soluble par un modèle **simple**.

Grâce aux codage, l'étude des classes de problèmes se ramène à l'étude des classes de langages.
Comparer les modèles de calcul du point de vue des langages qu'ils peuvent définir permet de savoir quels problèmes ils peuvent résoudre.

# Plan du cours

1) Introduction et théorie des automates
2) Machines de Turing
3) Langages reconnaissables et fonctions calculables
   **feat.** interro + sujet dm !
4) Langages décidables et machine universelle
5) Problèmes indécidables et théorème de Rice
   DM à rendre pour le 22 décembre (avant les vacances)
6) Révisions, et possiblement une jolie histoire

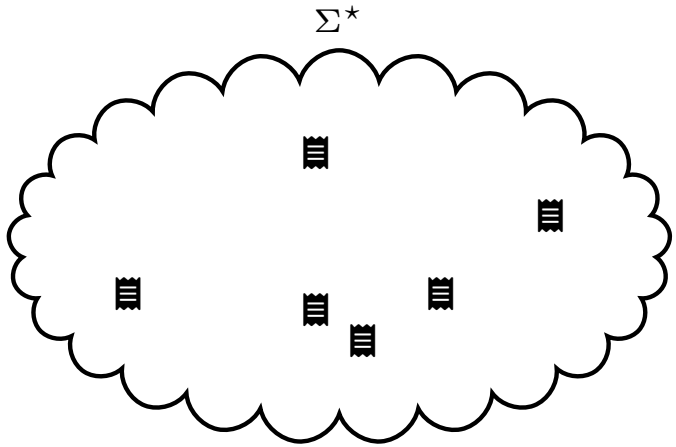
# Introduction
## Plan

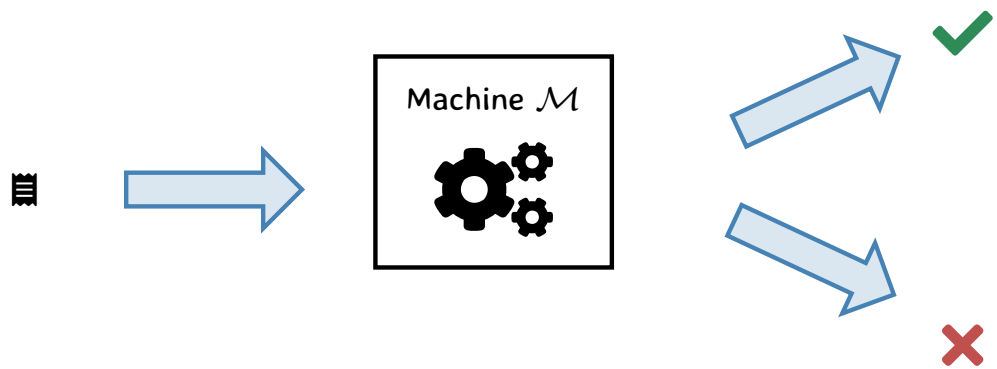
1. Problèmes de décision

2. Modèles de calcul

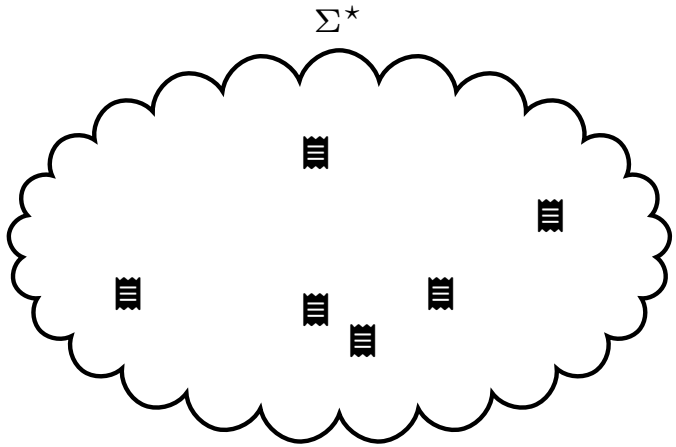
3. Théorie des automates

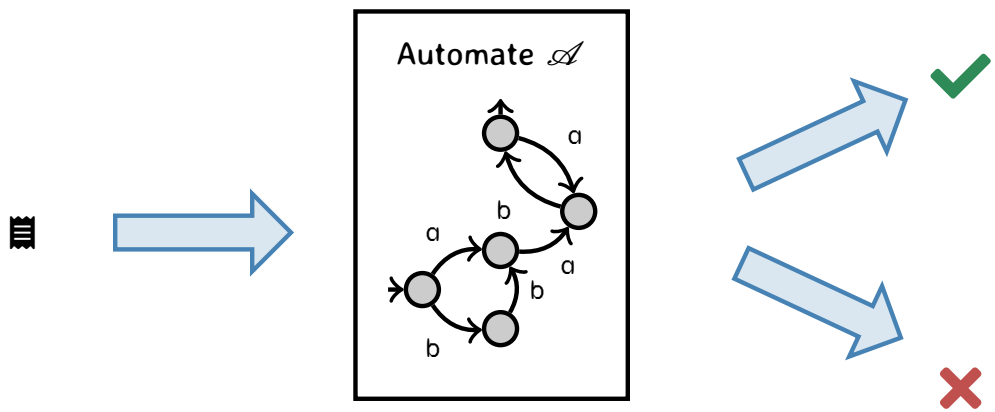
# Modèle de calcul
## Aujourd'hui : les automates



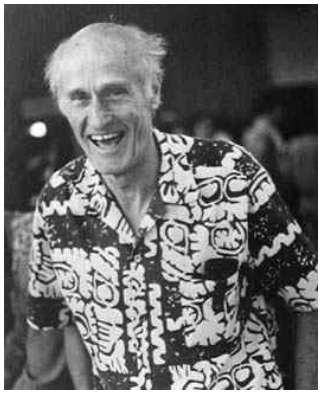



# Modèle de calcul
## Aujourd'hui : les automates



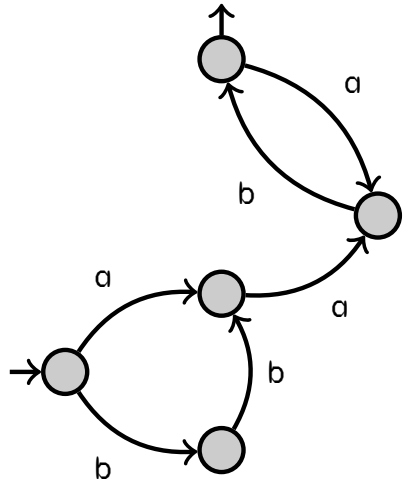



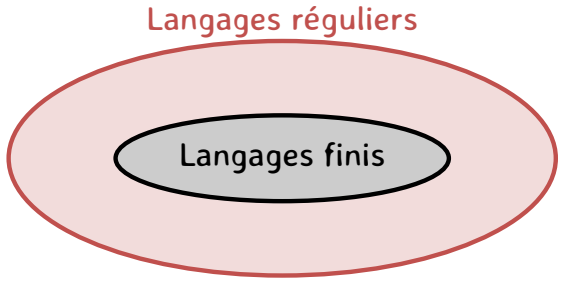
# Classes de langages



Stephen Cole Kleene

## Automates




## Expressions régulières

$$(a|bb)\ ab\ (ab)^\star$$

# Automates à états finis

Stephen Cole Kleene, “Representation of Events in Nerve Nets and Finite Automata”, Memorandum, Rand Corporation, 1951.

définition

Un **automate** est une structure $\mathscr{A} := \langle Q, \Sigma, \Delta, I, F \rangle$ où :

- $Q$ est un ensemble fini d'états ;
- $\Sigma$ est un alphabet (fini) ;
- $\Delta \subseteq Q \times \Sigma \times Q$ est un ensemble (fini) de transitions $p \xrightarrow{a} q$ ;
- $I, F \subseteq Q$ sont respectivement les ensembles d'états initiaux et finaux.

Une exécution de $\mathscr{A}$ sur le mot $w = w_1 \dots w_n \in \Sigma^\star$ est une séquence $q_0, \dots, q_n$ telle que $\forall 1 \leqslant i \leqslant n$, on a $q_{i-1} \xrightarrow{w_i} q_i \in \Delta$.
L'exécution est acceptante si $q_0 \in I$ et $q_n \in F$.
Le langage reconnu par $\mathscr{A}$ est l'ensemble $\mathcal{L}(\mathscr{A})$ des mots $w$ tels qu'il existe une exécution acceptante de $\mathscr{A}$ sur $w$.

# Langages réguliers

définition

Un langage $L$ est **régulier** si il existe un automate $\mathscr{A}$ tel que $\mathcal{L}(\mathscr{A}) = L$.

| Prouver qu'un langage est régulier | Prouver qu'un langage n'est pas régulier |
|---|---|
| exhiber un automate | ?? |

# Déterminisme

définition

Un automate est déterministe si

1) il a un seul état initial, i.e. $|I| = 1$ ;
2) il existe une fonction $\delta : Q \times \Sigma \to Q$ telle que

$$p \xrightarrow{a} q \in \Delta \Leftrightarrow q = \delta(p, a).$$

théorème

Pour tout langage régulier $L$, il existe un automate déterministe reconnaissant $L$.

**Preuve.** Déterminisation avec l'automate des parties. □

# Théorème de Kleene

Une expression régulière sur l'alphabet $\Sigma$ est un terme généré par la grammaire suivante :

$$e, f \in \mathrm{Reg}(\Sigma) ::= a \in \Sigma \mid \emptyset \mid \varepsilon \mid e \cdot f \mid e \cup f \mid e^\star.$$

Le langage dénoté par une expression est défini par induction :

$$[\![a]\!] = \{a\} \qquad [\![\emptyset]\!] = \emptyset \qquad [\![\varepsilon]\!] = \{\varepsilon\} \qquad [\![e \cup f]\!] = [\![e]\!] \cup [\![f]\!]$$

$$[\![e \cdot f]\!] = [\![e]\!] \cdot [\![f]\!] = \{uv \mid u \in [\![e]\!], v \in [\![f]\!]\} \qquad [\![e^\star]\!] = [\![e]\!]^\star = \{u_1 \cdots u_n \mid \forall i, u_i \in [\![e]\!]\}$$

théorème

Soit $\Sigma$ un alphabet fini et $L \subseteq \Sigma^\star$ un langage, alors les propriétés suivantes sont équivalentes :

- ☞ il existe un automate $\mathscr{A}$ tel que $\mathcal{L}(\mathscr{A}) = L$ ;
- ☞ il existe une expression $e \in \mathrm{Reg}(\Sigma)$ telle que $[\![e]\!] = L$.

# Théorème de Kleene

**Preuve.**

- ☞ Produire un automate à partir d'une expression : utiliser les propriétés de clôture des langages réguliers.
- ☞ Sens inverse : automates généralisés et élimination des états.

□

# Comment prouver qu'un langage n'est pas régulier

☞ On va prouver un invariant des langages réguliers.

# Comment prouver qu'un langage n'est pas régulier

- On va prouver un invariant des langages réguliers.
- Pour montrer qu'un langage n'est pas régulier, il suffit de montrer qu'il ne satisfait pas l'invariant.

# Lemme de l'étoile
## Idée

☞ Si un langage est régulier, alors il existe un automate qui le reconnait.

# Lemme de l'étoile
## Idée

- ☞ Si un langage est régulier, alors il existe un automate qui le reconnait.
- ☞ Cet automate à un nombre fini d'états. Notons $N := |Q|$.

# Lemme de l'étoile
## Idée

- Si un langage est régulier, alors il existe un automate qui le reconnait.
- Cet automate à un nombre fini d'états. Notons $N := |Q|$.
- Au cours d'une exécution suffisament longue, on va visiter le même état plusieurs fois.

# Lemme de l'étoile
## Idée

- Si un langage est régulier, alors il existe un automate qui le reconnait.
- Cet automate à un nombre fini d'états. Notons $N := |Q|$.
- Au cours d'une exécution suffisament longue, on va visiter le même état plusieurs fois.
- Si on a un fragment d'exécution $q \xrightarrow{w}{}^{*} q$, alors on peut le répéter autant de fois qu'on le souhaite, ce qui donne pour tout $n \in \mathbb{N}$ des exécutions :

$$q \xrightarrow{w^n}{}^{*} q.$$

# Énoncé

## Lemme de l'étoile [a]

a. En anglais : *Pumping lemma*.

Pour tout langage régulier $L$, il existe un entier $N \in \mathbb{N}$ tel que pour tout mot $w \in L$, si $|w| \geqslant N$, alors il existe des mots $u_1, u_2, u_3$ tels que :

- $w = u_1 u_2 u_3$, $u_2 \neq \varepsilon$, et $|u_1 u_2| \leqslant N$ ;
- $u_1 (u_2)^\star u_3 \subseteq L$, c'est à dire que pour tout entier $k \in \mathbb{N}$ on a $u_1 (u_2)^k u_3 \in L$.

# Lemme de l'étoile

**Preuve.** Soit $L$ régulier, et $\mathscr{A}$ tel que $\mathcal{L}(\mathscr{A}) = L$. On fixe $N := |Q|$.
Soit $w = w_1 \dots w_n \in \Sigma^\star$ tel que $w \in L$ et $|w| \geqslant N$, c'est à dire $n \geqslant N$.
On a une séquence d'états $(q_i)_{0 \leqslant i \leqslant n} \in Q^{n+1}$ tels que $q_0 \in I$, $q_n \in F$, et $\forall i,\ q_{i-1} \xrightarrow{w_i} q_i$, soit :

$$q_0 \xrightarrow{w_1} q_1 \xrightarrow{w_2} \dots \xrightarrow{w_N} q_N \dots \xrightarrow{w_n} q_n.$$

Comme $Q$ contient $N$ états différents, au cours des $N$ premières étapes de l'exécution, lorsqu'on visite les états $q_0, q_1, \dots, q_N$, il y a un état visité deux fois.
Donc il existe une paire d'indices $i < j \leqslant N$ tels que $q_i = q_j$.
On décompose $w$ comme suit :

| $w_1$ | $\dots$ | $w_i$ | $w_{i+1}$ | $\dots$ | $w_j$ | $w_{j+1}$ | $\dots$ | $w_n$ |
|---|---|---|---|---|---|---|---|---|
| $u_1$ | | | $u_2$ | | | $u_3$ | | |

En reportant cette décomposition sur l'exécution on obtient :

$$q_0 \xrightarrow{u_1}{}^* q_i \xrightarrow{u_2}{}^* q_j \xrightarrow{u_3}{}^* q_n$$

Comme $q_i = q_j$, cela signifie que $q_i \xrightarrow{u_2}{}^* q_i$, et donc que $\forall k \in \mathbb{N}$, on a $q_i \xrightarrow{u_2^k}{}^* q_i$, soit :

$$q_0 \in I \xrightarrow{u_1}{}^* q_i \xrightarrow{u_2^k}{}^* q_i = q_j \xrightarrow{u_3}{}^* q_n \in F.$$

D'où $u_1 (u_2)^\star u_3 \subseteq \mathcal{L}(\mathscr{A}) = L$. □

# Utilisation du lemme de l'étoile

Pour prouver qu'un langage $L$ n'est pas régulier, il suffit de trouver une fonction $\varphi : \mathbb{N} \to \Sigma^\star$ telle que :

☞ $\forall n \in \mathbb{N},\ |\varphi(n)| \geqslant n$ et $\varphi(n) \in L$ ;

☞ $\forall u_1, u_2, u_3 \in \Sigma^\star,\ \begin{cases} \varphi(n) = u_1 u_2 u_3 \\ u_2 \neq \varepsilon \\ |u_1 u_2| \leqslant n \end{cases} \Rightarrow \exists k \in \mathbb{N} :\ u_1 (u_2)^k u_3 \notin L.$

# Réduction

Considérons le langage $L_5$ sur l'alphabet $\{\texttt{open}, \texttt{close}\}$, constitué des mots bien parenthèsés :

$$\varepsilon \in L_5 \qquad u \in L_5 \Rightarrow \texttt{open} \cdot u \cdot \texttt{close} \in L_5 \qquad u, v \in L_5 \Rightarrow u \cdot v \in L_5.$$

# Réduction

Considérons le langage $L_5$ sur l'alphabet $\{\texttt{open}, \texttt{close}\}$, constitué des mots bien parenthèsés :

$$\varepsilon \in L_5 \qquad u \in L_5 \Rightarrow \texttt{open} \cdot u \cdot \texttt{close} \in L_5 \qquad u, v \in L_5 \Rightarrow u \cdot v \in L_5.$$

On remarque que $L_5 \cap (\texttt{open}^\star \texttt{close}^\star) = \{\texttt{open}^n \texttt{close}^n \mid n \in \mathbb{N}\} \simeq L_1$.
Si $L_5$ était régulier :

☞ il existe un automate $\mathscr{A}_5$ qui reconnait $L_5$ ;

# Réduction

Considérons le langage $L_5$ sur l'alphabet $\{\texttt{open}, \texttt{close}\}$, constitué des mots bien parenthèsés :

$$\varepsilon \in L_5 \qquad u \in L_5 \Rightarrow \texttt{open} \cdot u \cdot \texttt{close} \in L_5 \qquad u, v \in L_5 \Rightarrow u \cdot v \in L_5.$$

On remarque que $L_5 \cap (\texttt{open}^\star\texttt{close}^\star) = \{\texttt{open}^n\texttt{close}^n \mid n \in \mathbb{N}\} \simeq L_1$.
Si $L_5$ était régulier :

- il existe un automate $\mathscr{A}_5$ qui reconnait $L_5$ ;
- d'après le Théorème de Kleene, comme $\texttt{open}^\star\texttt{close}^\star$ est représentable par une expression régulière, il est régulier : il existe un automate $\mathscr{A}$ qui le reconnait ;

# Réduction

Considérons le langage $L_5$ sur l'alphabet $\{\texttt{open}, \texttt{close}\}$, constitué des mots bien parenthèsés :

$$\varepsilon \in L_5 \qquad u \in L_5 \Rightarrow \texttt{open} \cdot u \cdot \texttt{close} \in L_5 \qquad u, v \in L_5 \Rightarrow u \cdot v \in L_5.$$

On remarque que $L_5 \cap (\texttt{open}^\star\texttt{close}^\star) = \{\texttt{open}^n\texttt{close}^n \mid n \in \mathbb{N}\} \simeq L_1$.
Si $L_5$ était régulier :

- il existe un automate $\mathscr{A}_5$ qui reconnait $L_5$ ;
- d'après le Théorème de Kleene, comme $\texttt{open}^\star\texttt{close}^\star$ est représentable par une expression régulière, il est régulier : il existe un automate $\mathscr{A}$ qui le reconnait ;
- on sait calculer un automate qui reconnait l'intersection des langages de deux automates ;

# Réduction

Considérons le langage $L_5$ sur l'alphabet $\{\texttt{open}, \texttt{close}\}$, constitué des mots bien parenthèsés :

$$\varepsilon \in L_5 \qquad u \in L_5 \Rightarrow \texttt{open} \cdot u \cdot \texttt{close} \in L_5 \qquad u, v \in L_5 \Rightarrow u \cdot v \in L_5.$$

On remarque que $L_5 \cap (\texttt{open}^\star\texttt{close}^\star) = \{\texttt{open}^n\texttt{close}^n \mid n \in \mathbb{N}\} \simeq L_1$.
Si $L_5$ était régulier :

- il existe un automate $\mathscr{A}_5$ qui reconnait $L_5$ ;
- d'après le Théorème de Kleene, comme $\texttt{open}^\star\texttt{close}^\star$ est représentable par une expression régulière, il est régulier : il existe un automate $\mathscr{A}$ qui le reconnait ;
- on sait calculer un automate qui reconnait l'intersection des langages de deux automates ;
- donc on peut construire un automate $\mathscr{A}'$ tel que $\mathcal{L}(\mathscr{A}') = \mathcal{L}(\mathscr{A}_5) \cap \mathcal{L}(\mathscr{A})$ ;

# Réduction

Considérons le langage $L_5$ sur l'alphabet $\{\texttt{open}, \texttt{close}\}$, constitué des mots bien parenthèsés :

$$\varepsilon \in L_5 \qquad u \in L_5 \Rightarrow \texttt{open} \cdot u \cdot \texttt{close} \in L_5 \qquad u, v \in L_5 \Rightarrow u \cdot v \in L_5.$$

On remarque que $L_5 \cap (\texttt{open}^\star\texttt{close}^\star) = \{\texttt{open}^n\texttt{close}^n \mid n \in \mathbb{N}\} \simeq L_1$.
Si $L_5$ était régulier :

- il existe un automate $\mathscr{A}_5$ qui reconnait $L_5$ ;
- d'après le Théorème de Kleene, comme $\texttt{open}^\star\texttt{close}^\star$ est représentable par une expression régulière, il est régulier : il existe un automate $\mathscr{A}$ qui le reconnait ;
- on sait calculer un automate qui reconnait l'intersection des langages de deux automates ;
- donc on peut construire un automate $\mathscr{A}'$ tel que $\mathcal{L}(\mathscr{A}') = \mathcal{L}(\mathscr{A}_5) \cap \mathcal{L}(\mathscr{A})$ ;
- cet automate reconnait le language $\{\texttt{open}^n\texttt{close}^n \mid n \in \mathbb{N}\}$ ;

# Réduction

Considérons le langage $L_5$ sur l'alphabet $\{\texttt{open}, \texttt{close}\}$, constitué des mots bien parenthèsés :

$$\varepsilon \in L_5 \qquad u \in L_5 \Rightarrow \texttt{open} \cdot u \cdot \texttt{close} \in L_5 \qquad u, v \in L_5 \Rightarrow u \cdot v \in L_5.$$

On remarque que $L_5 \cap (\texttt{open}^\star\texttt{close}^\star) = \{\texttt{open}^n\texttt{close}^n \mid n \in \mathbb{N}\} \simeq L_1$.
Si $L_5$ était régulier :

- il existe un automate $\mathscr{A}_5$ qui reconnait $L_5$ ;
- d'après le Théorème de Kleene, comme $\texttt{open}^\star\texttt{close}^\star$ est représentable par une expression régulière, il est régulier : il existe un automate $\mathscr{A}$ qui le reconnait ;
- on sait calculer un automate qui reconnait l'intersection des langages de deux automates ;
- donc on peut construire un automate $\mathscr{A}'$ tel que $\mathcal{L}(\mathscr{A}') = \mathcal{L}(\mathscr{A}_5) \cap \mathcal{L}(\mathscr{A})$ ;
- cet automate reconnait le language $\{\texttt{open}^n\texttt{close}^n \mid n \in \mathbb{N}\}$ ;
- donc $L_1$ est régulier : contradiction !

# Réduction

Considérons le langage $L_5$ sur l'alphabet $\{\texttt{open}, \texttt{close}\}$, constitué des mots bien parenthèsés :

$$\varepsilon \in L_5 \qquad u \in L_5 \Rightarrow \texttt{open} \cdot u \cdot \texttt{close} \in L_5 \qquad u, v \in L_5 \Rightarrow u \cdot v \in L_5.$$

On remarque que $L_5 \cap (\texttt{open}^\star\texttt{close}^\star) = \{\texttt{open}^n\texttt{close}^n \mid n \in \mathbb{N}\} \simeq L_1$.
Si $L_5$ était régulier :

- il existe un automate $\mathscr{A}_5$ qui reconnait $L_5$ ;
- d'après le Théorème de Kleene, comme $\texttt{open}^\star\texttt{close}^\star$ est représentable par une expression régulière, il est régulier : il existe un automate $\mathscr{A}$ qui le reconnait ;
- on sait calculer un automate qui reconnait l'intersection des langages de deux automates ;
- donc on peut construire un automate $\mathscr{A}'$ tel que $\mathcal{L}(\mathscr{A}') = \mathcal{L}(\mathscr{A}_5) \cap \mathcal{L}(\mathscr{A})$ ;
- cet automate reconnait le language $\{\texttt{open}^n\texttt{close}^n \mid n \in \mathbb{N}\}$ ;
- donc $L_1$ est régulier : contradiction !

Schéma de preuve

Si $L_5$ est régulier, alor $L_1$ est régulier. Comme $L_1$ n'est pas régulier, $L_5$ ne l'est pas non plus.