IF228 - Calculabilité et Complexité

Cours #1 : problème, langage, machine de Turing

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

8 janvier 2025

Plan

1. Calculabilité
2. Problèmes de décision et langages
3. Machines de Turing

Organisation

Trivia

Intervenants :

- Frédéric Herbreteau (frederic.herbreteau@bordeaux-inp.fr)
- Thibault Hilaire thibault.hilaire@u-bordeaux.fr
- Noureddine Mouhoub noureddine.mouhoub@u-bordeaux.fr
- Géraud Senizergues (geraud.senizergues@u-bordeaux.fr)

Organisation :

- Cours : $8 \times 1h20$
- TDs : $8 \times 2h$

Contrôle-continu : 2 devoirs sur table individuels

- vendredi 14/02 de 16h15 à 17h15
- vendredi 28/03 de 16h15 à 17h15

Ressources

Supports de cours et de TDs, devoirs :

`https://thor.enseirb-matmeca.fr/sites/if228-class/`

Erreurs, oublis, critiques, contresens :

$\longrightarrow$ `frederic.herbreteau@bordeaux-inp.fr`

Livres :

- Sipser. **Introduction to the Theory of Computation**
- John Hopcroft, Rajeev Motwani, Jeffrey Ullman. **Introduction to Automata Theory, Languages, and Computation**
- Sanjeev Arora et Boaz Barak. **Computationnal Complexity A Modern Approach**

Introduction

Règle et compas

Règle et compas

- Outils : Règle non graduée et compas

Règle et compas

- Outils : Règle non graduée et compas
- But : construire des figures géométriques dans le plan

Modèle de calcul

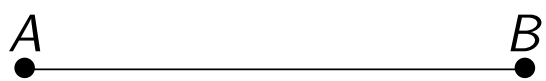
Initialement, 2 points du plan sont donnés
la distance entre eux définit l'unité

Règles :

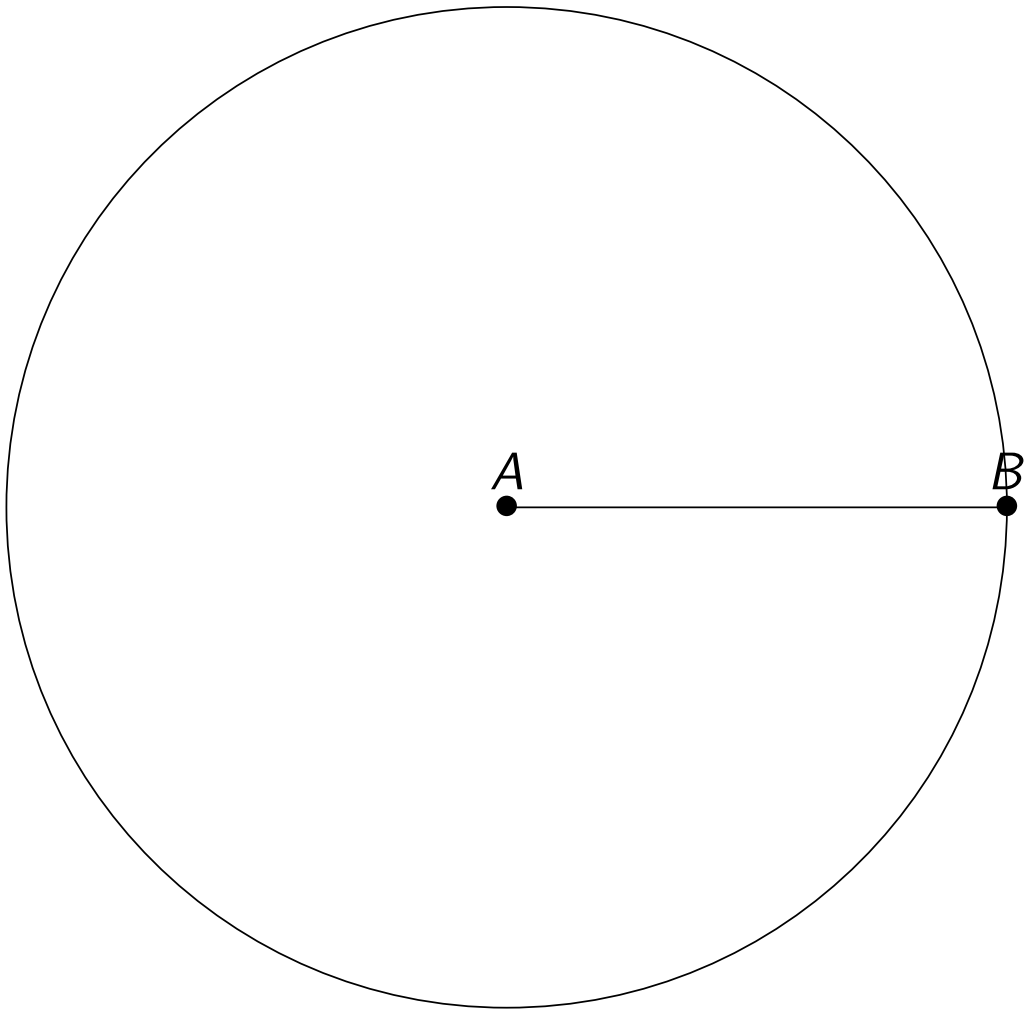
- Étant donné deux points A, B, tracer le segment [AB]
- Étant donné deux points A, B, tracer le cercle de centre A et de rayon AB
- Tracer un point à l'intersection de n'importe quelle paire d'objets précédemment construits

Exemple : médiatrice

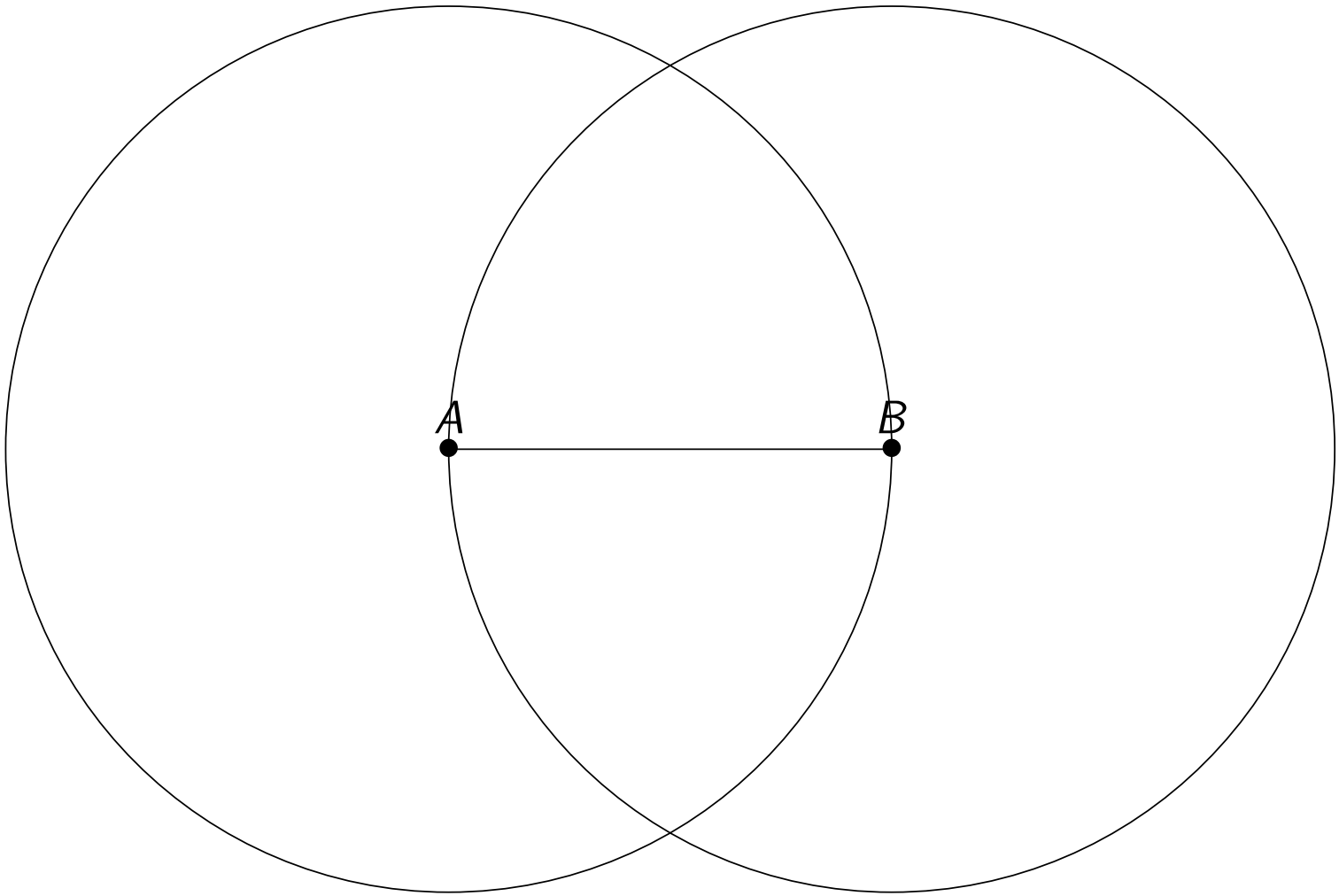
Exemple : médiatrice



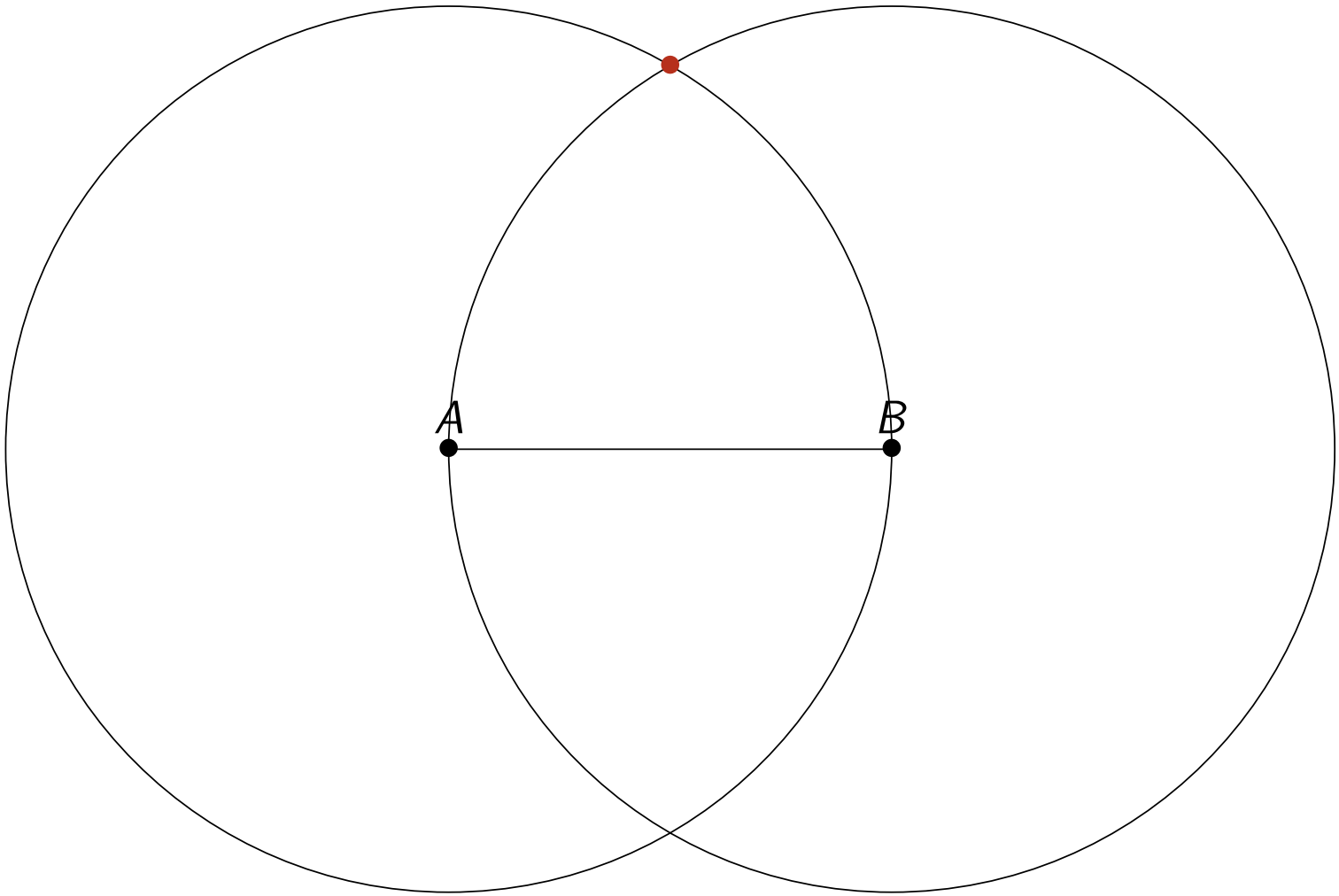
Exemple : médiatrice



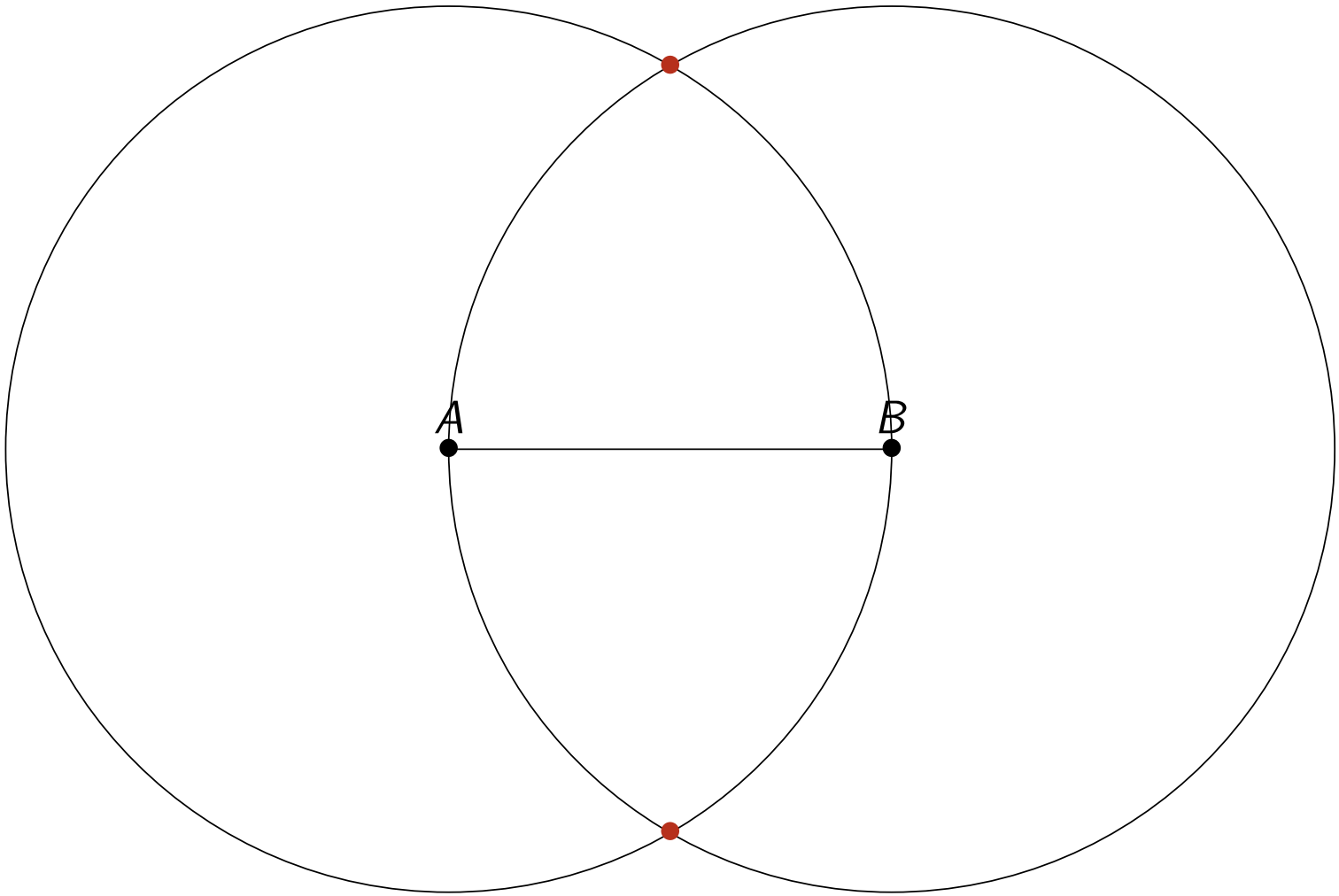
Exemple : médiatrice



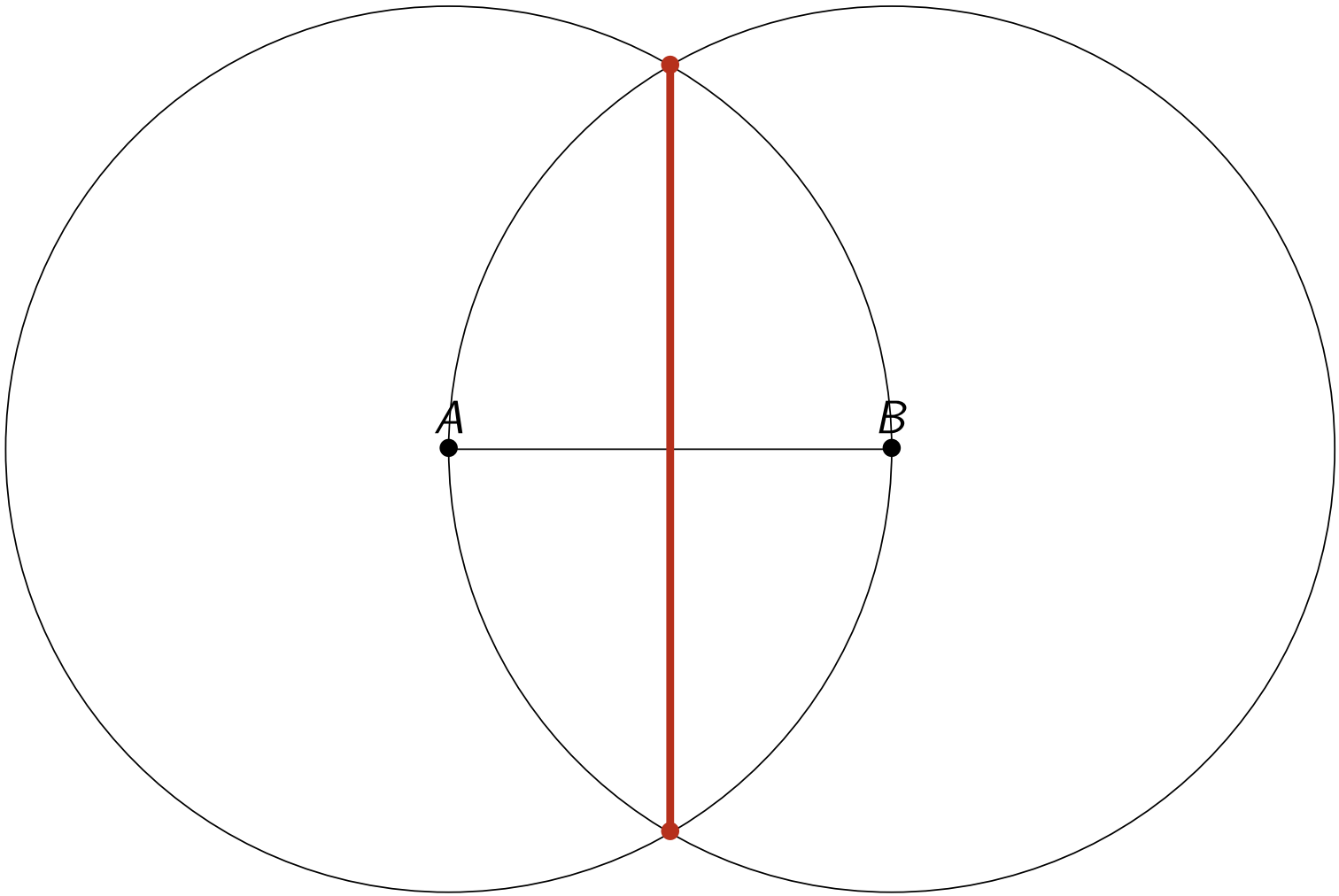
Exemple : médiatrice



Exemple : médiatrice



Exemple : médiatrice



Médiatrice

Problème : étant donné 2 points distincts A et B, tracer la **médiatrice** du segment [AB]

Médiatrice

Problème : étant donné 2 points distincts A et B, tracer la **médiatrice** du segment [AB]

1. tracer le cercle C de centre A et de rayon AB
2. tracer le cercle C' de centre B et de rayon AB
3. tracer la ligne passant par les intersections de C et C'

Médiatrice

Problème : étant donné 2 points distincts A et B, tracer la **médiatrice** du segment [AB]

Algorithme

1. tracer le cercle C de centre A et de rayon AB
2. tracer le cercle C' de centre B et de rayon AB
3. tracer la ligne passant par les intersections de C et C'

Question

Est-il possible de construire n'importe quelle figure géométrique à la règle et au compas ?

Question

Est-il possible de construire n'importe quelle figure géométrique à la règle et au compas ?

Autrement dit : Quelles sont les figures géométriques pour lesquelles il existe un algorithme qui les construit ?
Quelle est la puissance d'expression du modèle règle & compas ?

Algorithme : suite finie d'applications des règles du modèle de calcul

Question

Est-il possible de construire n'importe quelle figure géométrique à la règle et au compas ?

Autrement dit : Quelles sont les figures géométriques pour lesquelles il existe un algorithme qui les construit ?
Quelle est la puissance d'expression du modèle règle & compas ?

Algorithme : suite finie d'applications des règles du modèle de calcul

Existence ou non d'algorithmes : Calculabilité

Règles & compas : problèmes

- Construction de polygones : construire un pentagone, un heptagone, un heptadécagone (17 côtés), etc ?

Règles & compas : problèmes

- Construction de polygones : construire un pentagone, un heptagone, un heptadécagone (17 côtés), etc ?

- Trissection d'un angle : étant donné un angle, tracer les 2 segments qui le partagent en 3 secteurs égaux

Règles & compas : problèmes

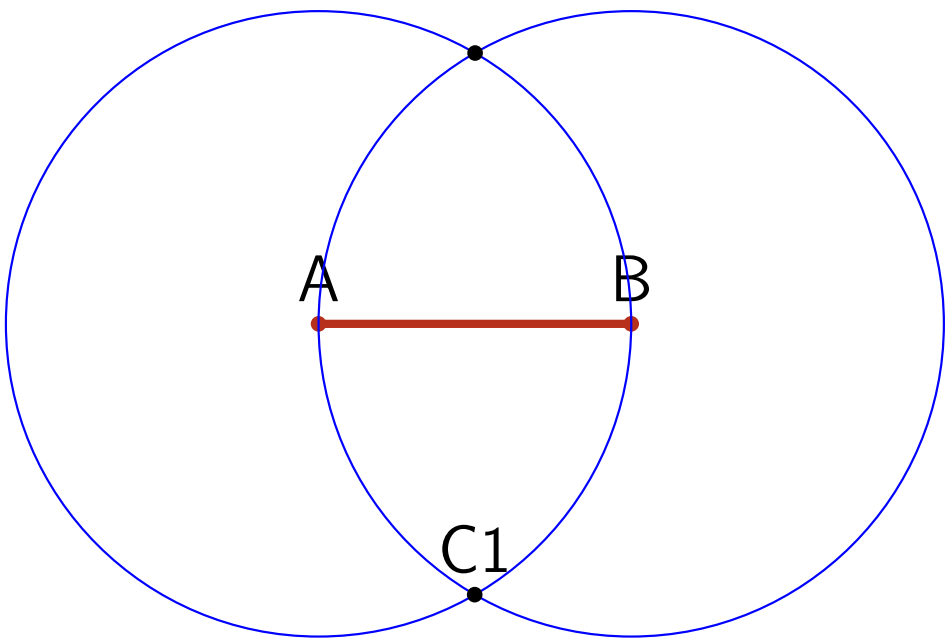
- Construction de polygones : construire un pentagone, un heptagone, un heptadécagone (17 côtés), etc ?

- Trissection d'un angle : étant donné un angle, tracer les 2 segments qui le partagent en 3 secteurs égaux

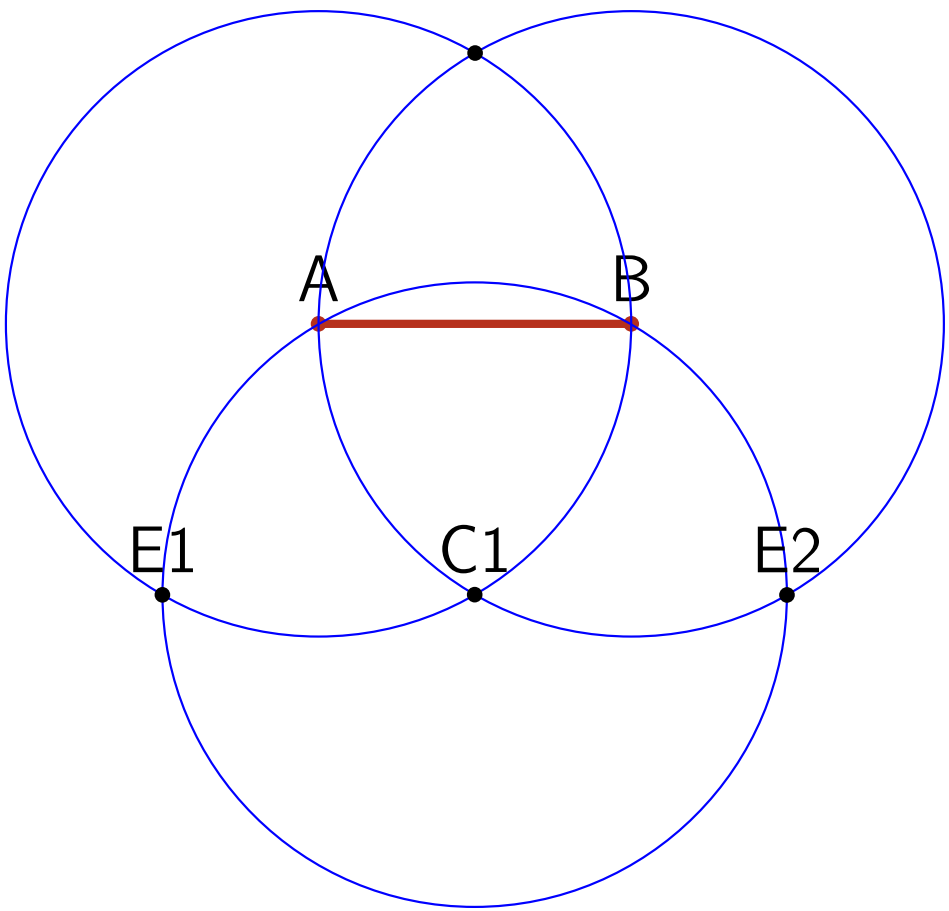
- Duplication du cube : étant donné l'arête d'un cube C, tracer l'arête du cube dont le volume est le double de celui de C

Pentagone

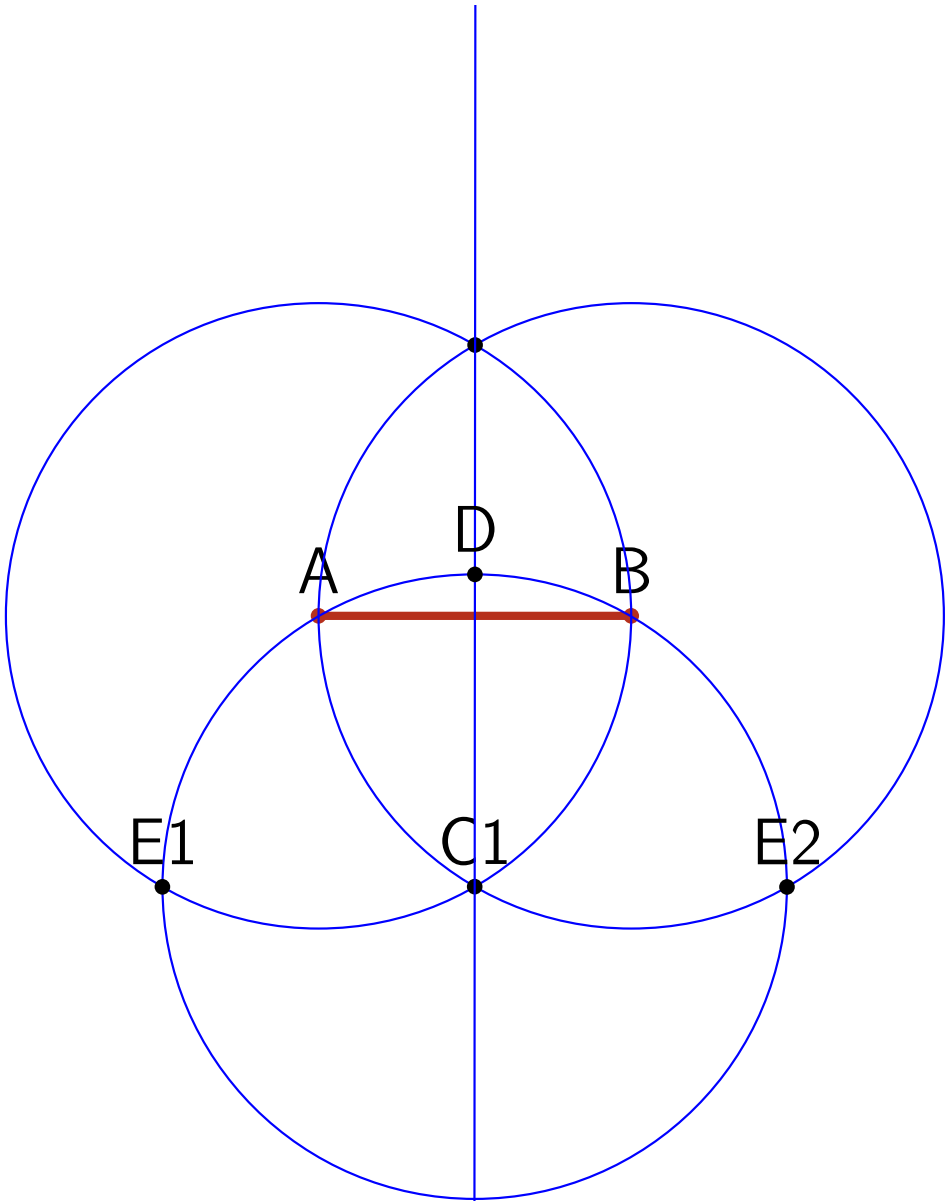
Pentagone



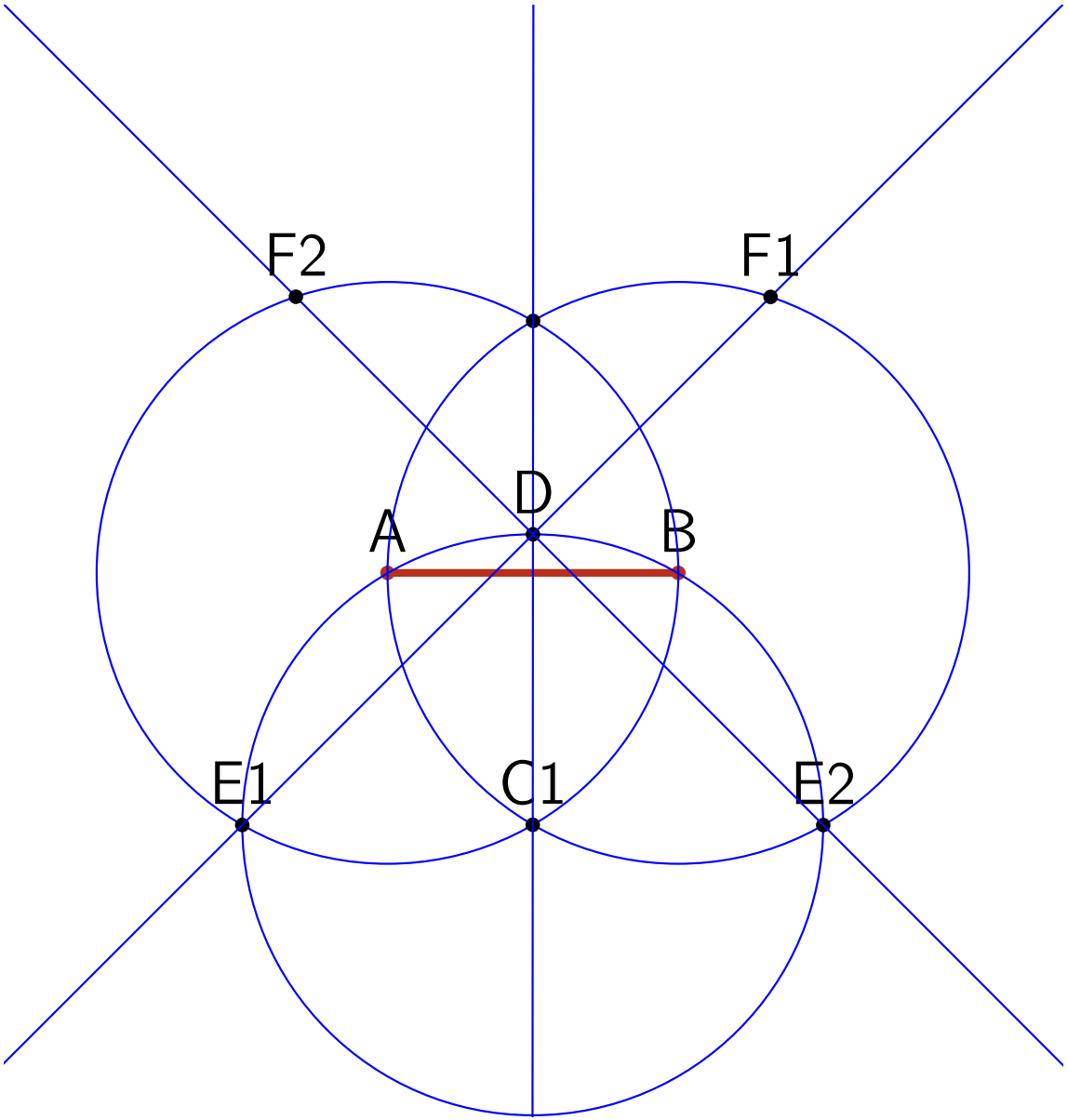
Pentagone



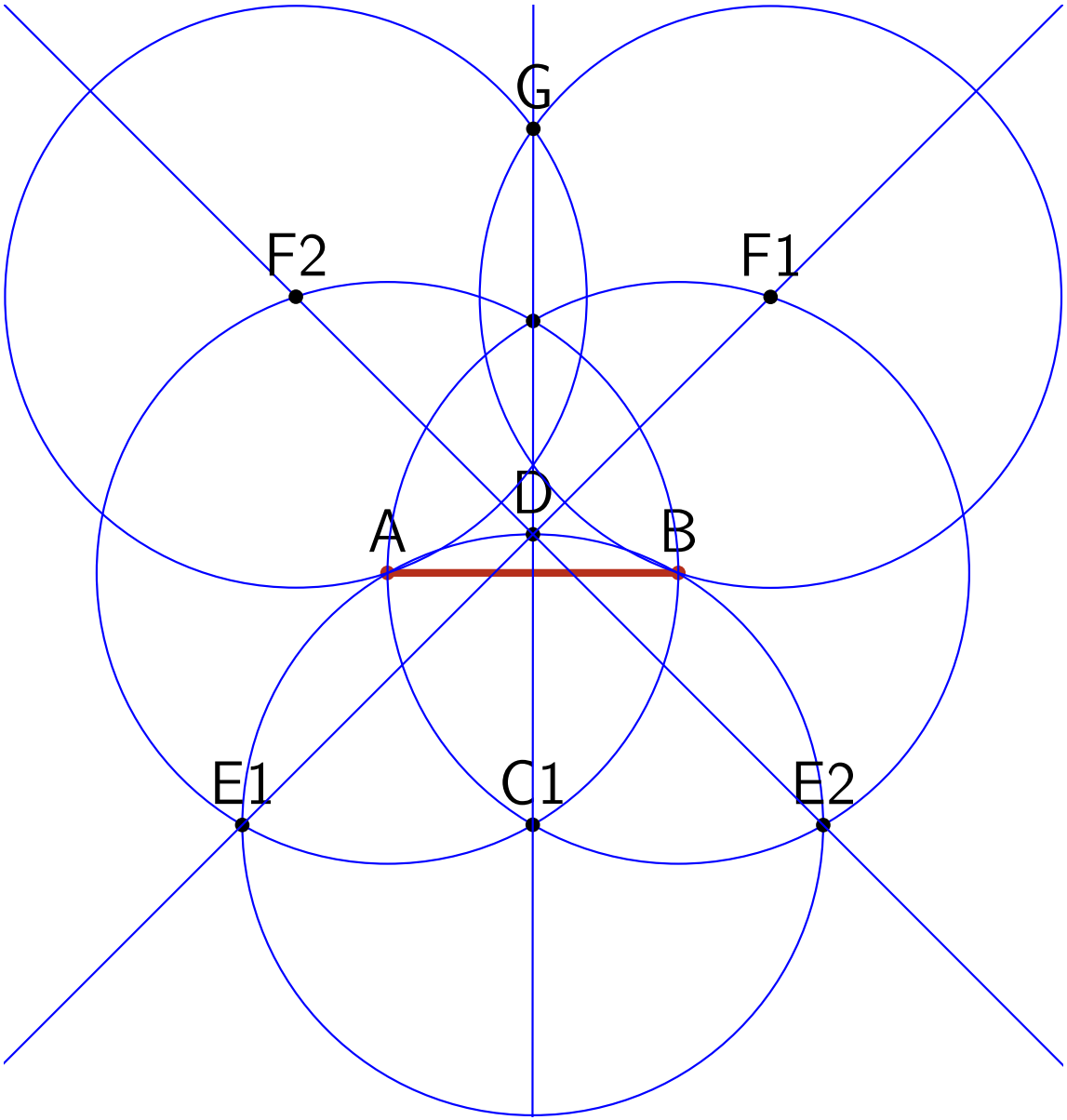
Pentagone



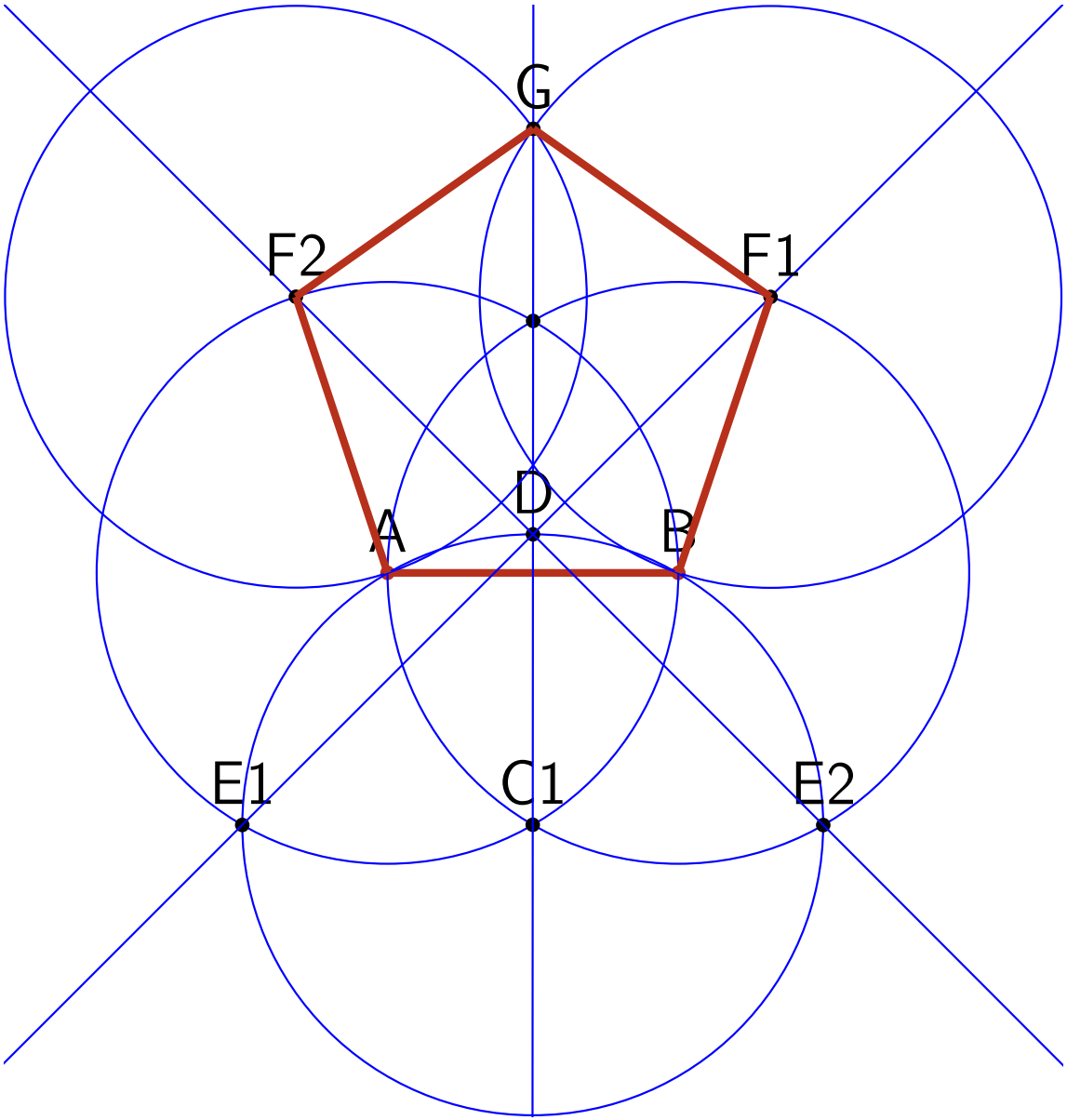
Pentagone



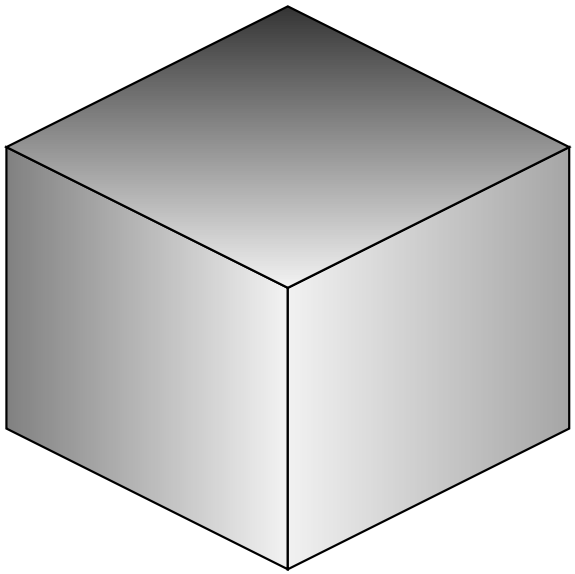
Pentagone



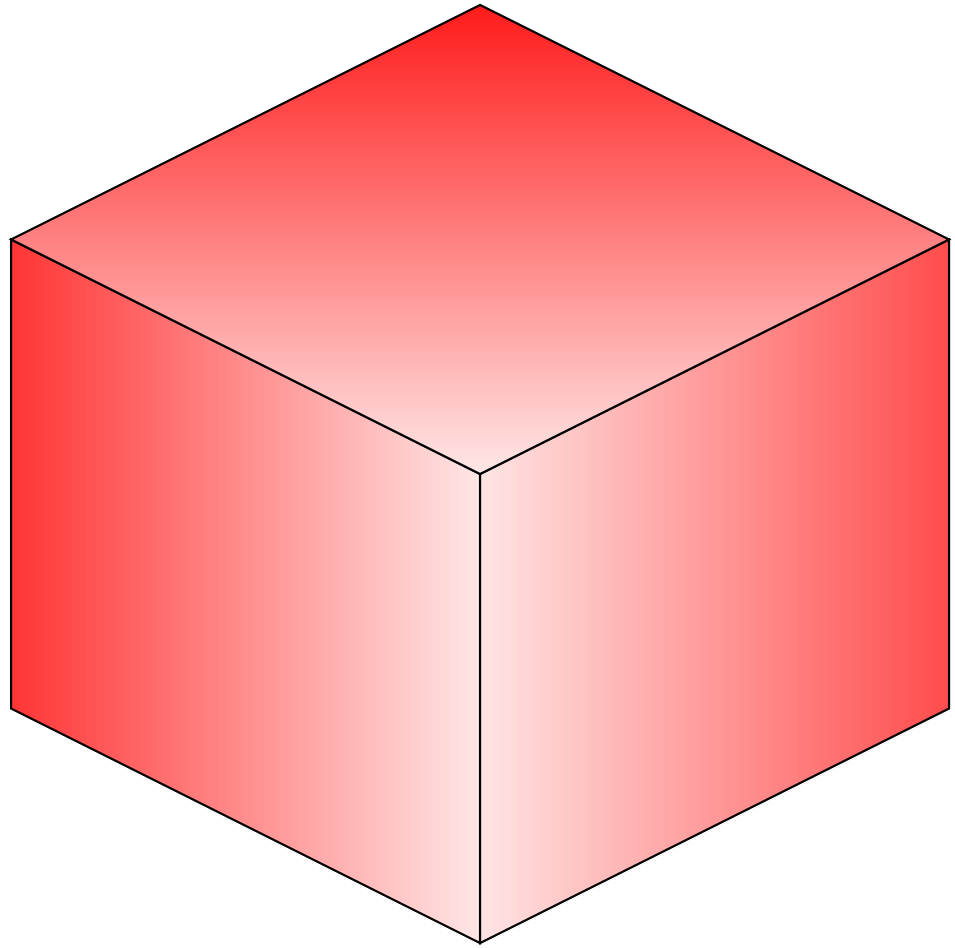
Pentagone



Duplication du cube

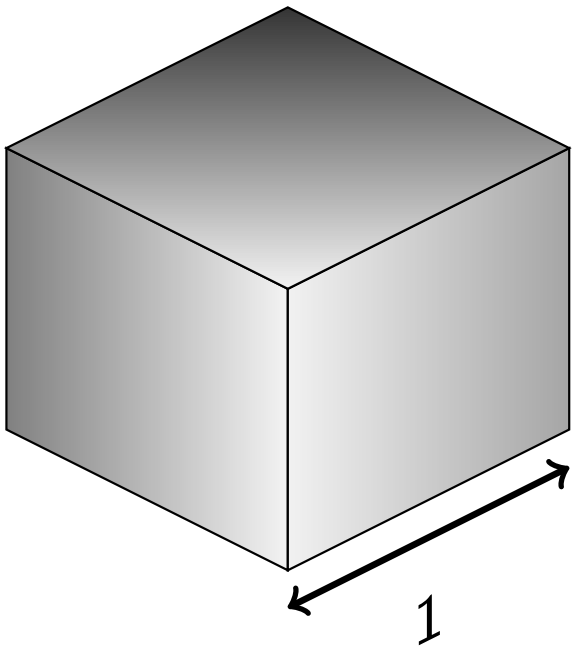


Volume V

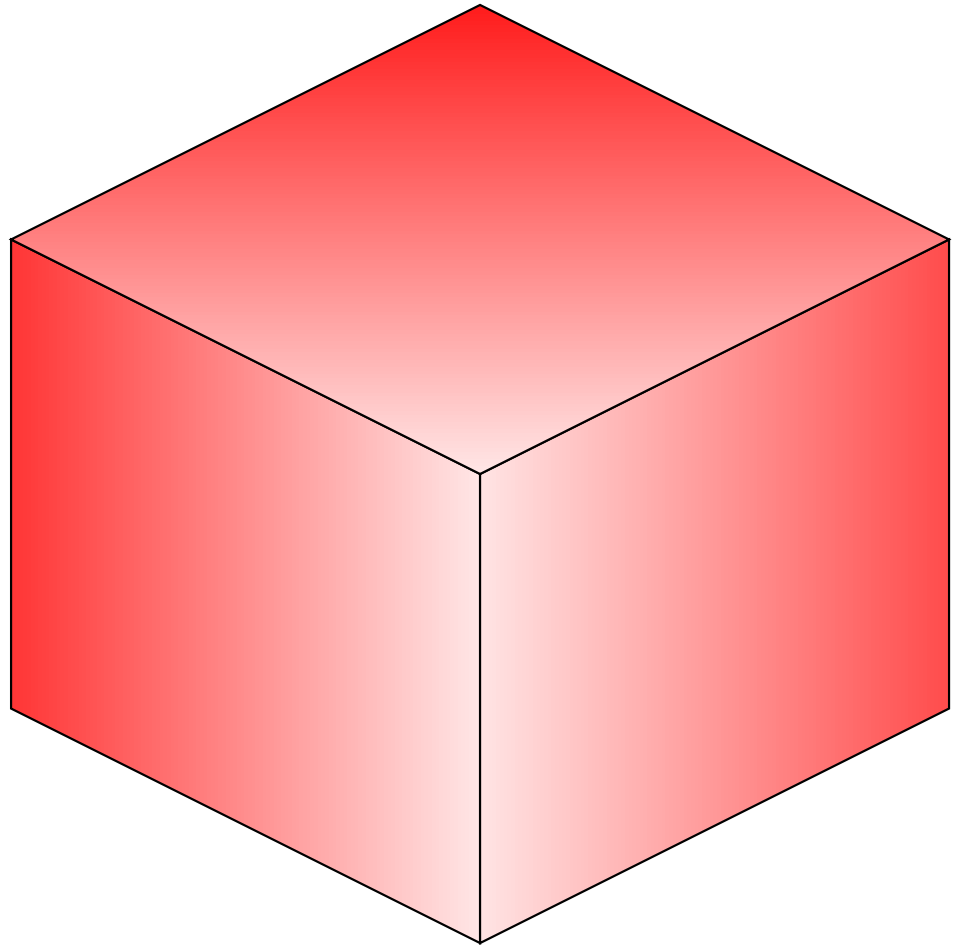


Volume 2V

Duplication du cube

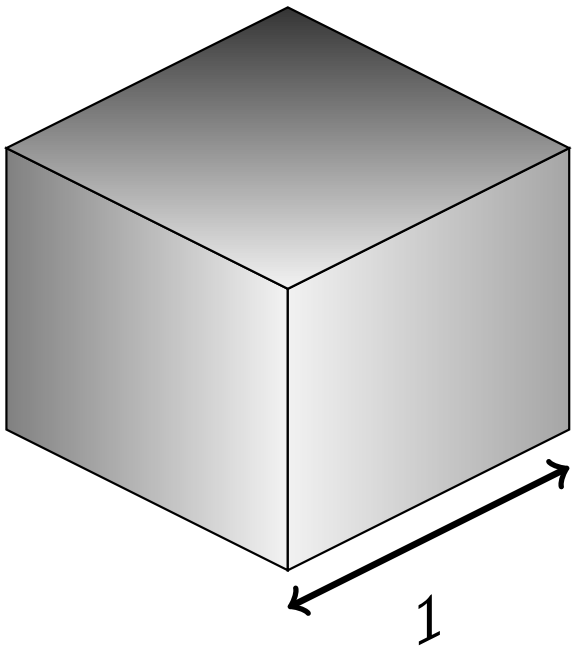


Volume V

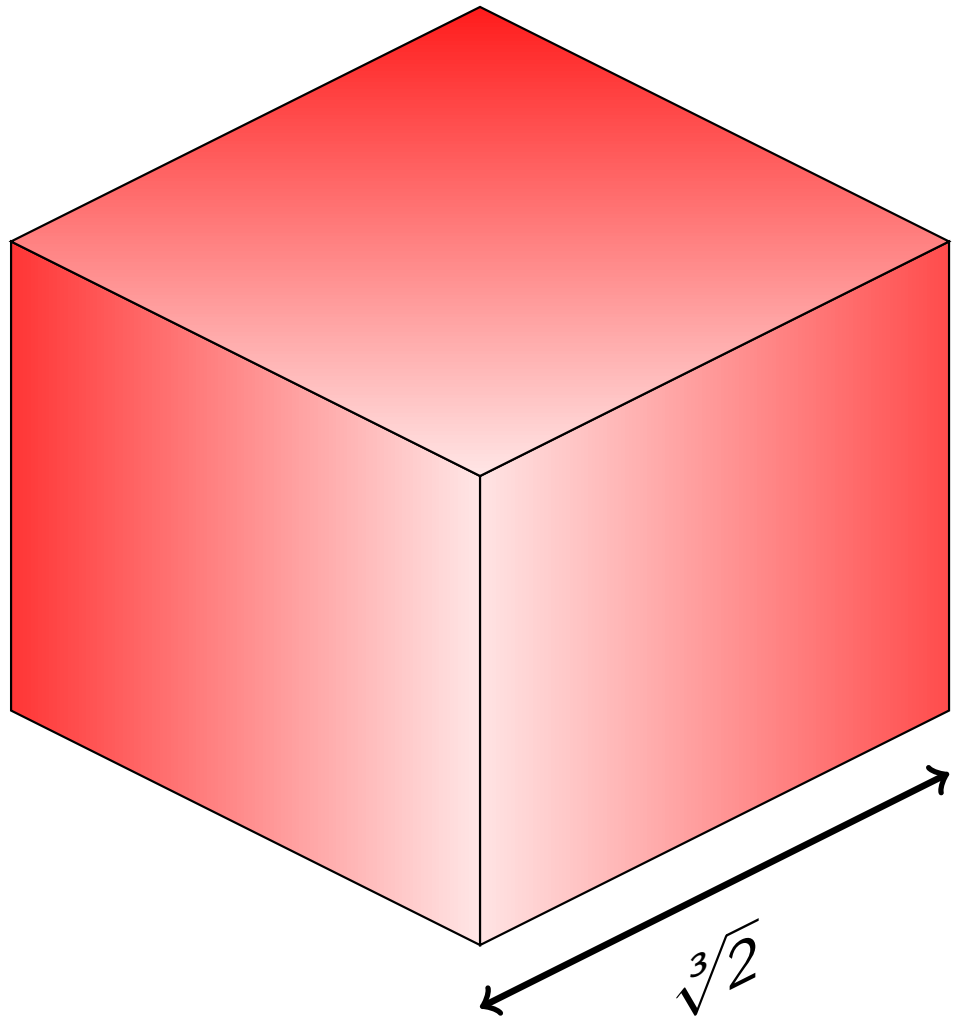


Volume 2V

Duplication du cube



Volume V



Volume 2V

Duplication du cube

- Longueur arête du cube $C = 1$
- Longueur arête du cube de volume double $=$

Duplication du cube

- Longueur arête du cube $C = 1$
- Longueur arête du cube de volume double $= \sqrt[3]{2}$

Duplication du cube

- Longueur arête du cube $C = 1$
- Longueur arête du cube de volume double = $\sqrt[3]{2}$

Problème équivalent : étant donné un segment de longueur 1, tracer un segment de longueur $\sqrt[3]{2}$

Duplication du cube

- Longueur arête du cube $C = 1$
- Longueur arête du cube de volume double $= \sqrt[3]{2}$

Problème équivalent : étant donné un segment de longueur 1, tracer un segment de longueur $\sqrt[3]{2}$

Existe-t-il un algorithme dans le modèle règle & compas pour résoudre le problème de la duplication du cube ?

Duplication du cube

- XVIIème : coordonnées cartésiennes
  - la droite passant par (a, b) et (c, d) : $y = \frac{b-d}{a-c}x + \frac{da-bc}{a-c}$
  - le cercle de centre (a, b) et passant par (c, d) : $(x-a)^2 + (y-b)^2 = (a-c)^2 + (b-d)^2$

- coordonnées d'intersection :
  - droite/droite $\longrightarrow$ équation linéaire
  - droite/cercle ou cercle/cercle $\longrightarrow$ équation quadratique

Duplication du cube

- XVIIème : coordonnées cartésiennes
  - la droite passant par (a, b) et (c, d) : $y = \frac{b-d}{a-c}x + \frac{da-bc}{a-c}$
  - le cercle de centre (a, b) et passant par (c, d) : $(x-a)^2 + (y-b)^2 = (a-c)^2 + (b-d)^2$

- coordonnées d'intersection :
  - droite/droite $\longrightarrow$ équation linéaire
  - droite/cercle ou cercle/cercle $\longrightarrow$ équation quadratique

Réduction : obtenir $\sqrt[3]{2}$ à partir des nombres 0 et 1 en utilisant uniquement les opérations $+, -, *, \div$ et $\sqrt{\ }$?

Duplication du cube

On peut montrer qu'il n'est pas possible d'obtenir $\sqrt[3]{2}$ à partir des nombres 0 et 1 en utilisant uniquement les opérations $+, -, *, \div$ et $\sqrt{\ }$

Duplication du cube

On peut montrer qu'il n'est pas possible d'obtenir $\sqrt[3]{2}$ à partir des nombres 0 et 1 en utilisant uniquement les opérations $+, -, *, \div$ et $\sqrt{\ }$

Conséquence : le problème de la duplication du cube n'a pas de solution dans le modèle règle & compas.

Duplication du cube

On peut montrer qu'il n'est pas possible d'obtenir $\sqrt[3]{2}$ à partir des nombres 0 et 1 en utilisant uniquement les opérations $+, -, *, \div$ et $\sqrt{\ }$

Conséquence : le problème de la duplication du cube n'a pas de solution dans le modèle règle & compas.

- Remarque : problème connu dès l'Antiquité
- Impossibilité : M.L Wantzel, "Recherches sur les moyens de reconnaître si un Problème de Géométrie peut se résoudre avec la règle et le compas", **1837**

Polygone régulier

Polygone régulier à n **cotés** constructible à la règle et au compas ssi n est le produit d'une puissance de 2 et de nombre premiers impairs distincts de Fermat (Wantzel 1837).

(NB : un nombre premier p est "de Fermat" ssi il est de la forme $2^{2^n} + 1$)

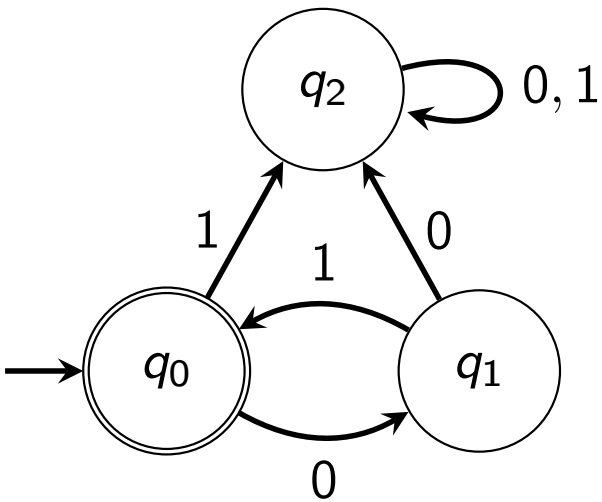
Automates finis

- Outils : Lire une lettre et mettre à jour une **mémoire finie** (registre)

Automates finis

- Outils : Lire une lettre et mettre à jour une **mémoire finie** (registre)

Problème 1 : accepter le langage $(01)^*$



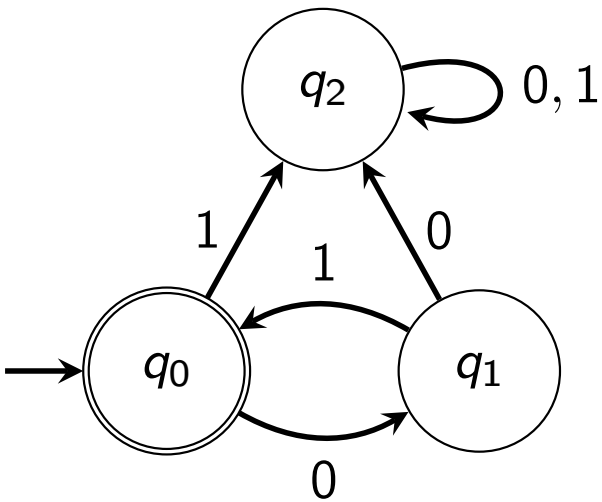
Instructions δ :

$$q_0, 0 \to q_1$$
$$q_0, 1 \to q_2$$
$$q_1, 0 \to q_2$$
$$q_1, 1 \to q_0$$
$$q_2, . \to q_2$$

Automates finis

- Outils : Lire une lettre et mettre à jour une **mémoire finie** (registre)

Problème 1 : accepter le langage $(01)^*$



Instructions δ :

$$q_0, 0 \to q_1$$
$$q_0, 1 \to q_2$$
$$q_1, 0 \to q_2$$
$$q_1, 1 \to q_0$$
$$q_2, . \to q_2$$

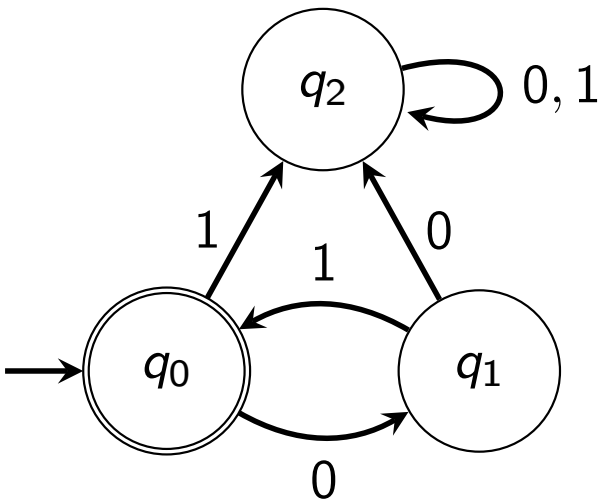
Problème 2 : accepter le langage des mots palindromes

$PAL = \{w \in \{0,1\}^* \mid w = miroir(w)\}$

Automates finis

- Outils : Lire une lettre et mettre à jour une **mémoire finie** (registre)

Problème 1 : accepter le langage $(01)^*$



Instructions δ :

$$q_0, 0 \to q_1$$
$$q_0, 1 \to q_2$$
$$q_1, 0 \to q_2$$
$$q_1, 1 \to q_0$$
$$q_2, . \to q_2$$

Problème 2 : accepter le langage des mots palindromes
$PAL = \{w \in \{0,1\}^* \mid w = miroir(w)\}$

$\longrightarrow$ **Impossible :** PAL n'est pas régulier

Calculabilité

Que peut-on calculer ?

règle & compas	**ordinateur**
n-gone régulier, trissection, duplication du cube, ..	plus court chemin, est-ce que ce programme s'arrête, est-ce que ce programme est correct ?..

Calculabilité

Que peut-on calculer ?

règle & compas	**ordinateur**
n-gone régulier, trissection, duplication du cube, ..	plus court chemin, est-ce que ce programme s'arrête, est-ce que ce programme est correct ?..

$\longrightarrow$ Modèle de calcul pour "l'ordinateur" ?

Langages, problèmes et fonctions

Rappel : mots

Un alphabet Σ est un ensemble fini de symboles, ex : $\Sigma = \{0, 1\}$.

Un mot sur l'alphabet Σ est une suite finie de symboles de Σ, ex : $w = 01010101010$.

On note ϵ le mot vide.

La longueur d'un mot w, noté $|w|$, est le nombre de symboles qui le composent. ex : $|w| = 11$ et $|\epsilon| = 0$.

Σ^* est l'ensemble des mots sur l'alphabet Σ. Un langage L sur l'alphabet Σ est une partie de Σ^*, i.e. $L \subseteq \Sigma^*$.

Problème

Un problème est la donnée de :

- un ensemble d'entrées $\mathcal{I}$ (peut être infini)
- un ensemble de sorties $\mathcal{O}$ (peut être infini)
- une relation $\Delta \subseteq \mathcal{I} \times \mathcal{O}$

Problème

Un problème est la donnée de :

- un ensemble d'entrées $\mathcal{I}$ (peut être infini)
- un ensemble de sorties $\mathcal{O}$ (peut être infini)
- une relation $\Delta \subseteq \mathcal{I} \times \mathcal{O}$

Si $(I, O) \in \Delta$, pour $I \in \mathcal{I}$ et $O \in \mathcal{O}$, alors O est solution du problème pour l'entrée I.

Problème

Un problème est la donnée de :

- un ensemble d'entrées $\mathcal{I}$ (peut être infini)
- un ensemble de sorties $\mathcal{O}$ (peut être infini)
- une relation $\Delta \subseteq \mathcal{I} \times \mathcal{O}$

Si $(I, O) \in \Delta$, pour $I \in \mathcal{I}$ et $O \in \mathcal{O}$, alors O est solution du problème pour l'entrée I.

Exemple : déterminer si un nombre est pair

- $\mathcal{I} =$
- $\mathcal{O} =$
- $\Delta =$

Fonctions et codage

On s'intéressera le plus souvent au calcul de fonctions

$$f : \{0,1\}^* \to \{0,1\}^*$$

Fonctions et codage

On s'intéressera le plus souvent au calcul de fonctions

$$f : \{0,1\}^* \to \{0,1\}^*$$

- On peut aisément encoder en binaire : les **entiers**, ***n*-uplets d'entiers**, **chaînes de caractères**, et les objets plus complexes comme les **graphes**, etc.

  ex : $34 \to 100010$, graphe $\to$ matrice(binaire) d'adjacence

Fonctions et codage

On s'intéressera le plus souvent au calcul de fonctions

$$f : \{0,1\}^* \to \{0,1\}^*$$

- On peut aisément encoder en binaire : les **entiers**, ***n*-uplets d'entiers**, **chaînes de caractères**, et les objets plus complexes comme les **graphes**, etc.

  ex : $34 \to 100010$, graphe $\to$ matrice(binaire) d'adjacence

$\Rightarrow$ Pas de perte de généralité

Fonctions et codage

On s'intéressera le plus souvent au calcul de fonctions

$$f : \{0,1\}^* \to \{0,1\}^*$$

- On peut aisément encoder en binaire : les **entiers**, *n*-**uplets d'entiers**, **chaînes de caractères**, et les objets plus complexes comme les **graphes**, etc.

  ex : $34 \to 100010$, graphe $\to$ matrice(binaire) d'adjacence

$\Rightarrow$ Pas de perte de généralité

Une fonction booléenne (ou prédicat) est une fonction

$$f : \{0,1\}^* \to \{0,1\}$$

Problème de décision/langage

Un problème $P = (\mathcal{I}, \mathcal{O}, \Delta)$ est un problème de décision si :

- l'ensemble des sorties $\mathcal{O} = \{0, 1\}$ (faux/vrai)
- et chaque entrée $I \in \mathcal{I}$, a **exactement une solution** : soit $(I, 0) \in \Delta$, soit $(I, 1) \in \Delta$

Problème de décision/langage

Un problème $P = (\mathcal{I}, \mathcal{O}, \Delta)$ est un problème de décision si :

- l'ensemble des sorties $\mathcal{O} = \{0, 1\}$ (faux/vrai)
- et chaque entrée $I \in \mathcal{I}$, a **exactement une solution** : soit $(I, 0) \in \Delta$, soit $(I, 1) \in \Delta$

Ainsi, à tout problème de décision, on peut associer :

- la fonction booléenne f_P :

$$f_P(I) = \begin{cases} 1 & \text{si } (I, 1) \in \Delta \\ 0 & \text{sinon} \end{cases}$$

- le langage L_P :

$$L_P = \{w \in \{0, 1\}^* \mid f_P(w) = 1\}$$

Palindrome

Soit Σ un alphabet.

- (informel) déterminer si un mot $w \in \Sigma$ est un palindrome
  ex :

Palindrome

Soit Σ un alphabet.

- (informel) déterminer si un mot $w \in \Sigma$ est un palindrome
  ex :

- (problème) $\mathcal{I} = \Sigma^*$, $\mathcal{O} = \{0, 1\}$,
  $(w, 1) \in \Delta \iff w$ est un palindrome
  ex :

Palindrome

Soit Σ un alphabet.

- (informel) déterminer si un mot $w \in \Sigma$ est un palindrome
  ex :
- (problème) $\mathcal{I} = \Sigma^*$, $\mathcal{O} = \{0, 1\}$,
  $(w, 1) \in \Delta \iff w$ est un palindrome
  ex :
- (langage) PAL $= \{w \in \Sigma^* : w = \text{mirroir}(w)\}$
  ex :

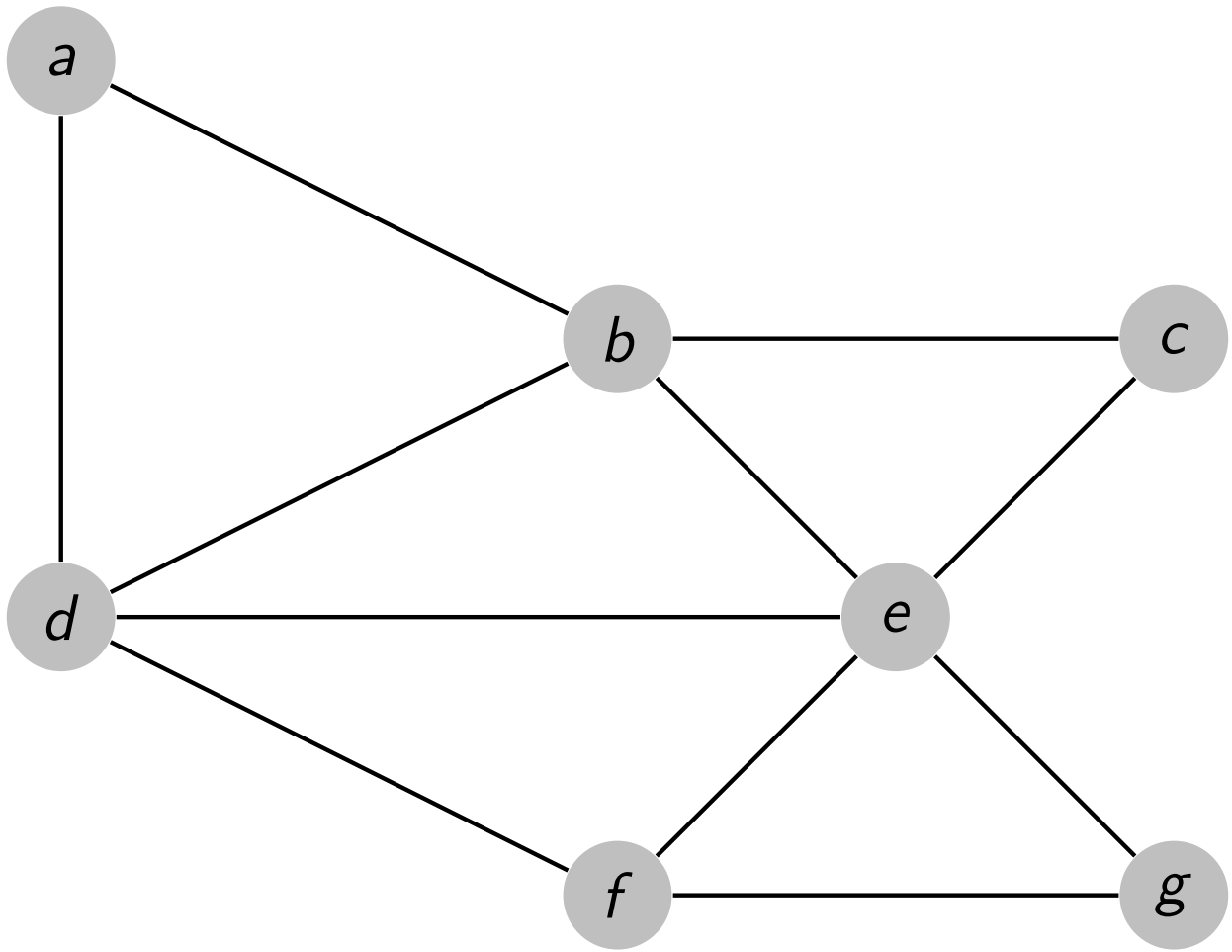
Palindrome

Soit Σ un alphabet.

- (informel) déterminer si un mot $w \in \Sigma$ est un palindrome
  ex :
- (problème) $\mathcal{I} = \Sigma^*$, $\mathcal{O} = \{0, 1\}$,
  $(w, 1) \in \Delta \iff w$ est un palindrome
  ex :
- (langage) $\text{PAL} = \{w \in \Sigma^* : w = \text{mirroir}(w)\}$
  ex :
- (fonction) $f_{\text{PAL}} : \Sigma^* \to \{0, 1\}$ telle que
  $f_{\text{PAL}}(w) = 1 \iff w$ est un palindrome
  ex :

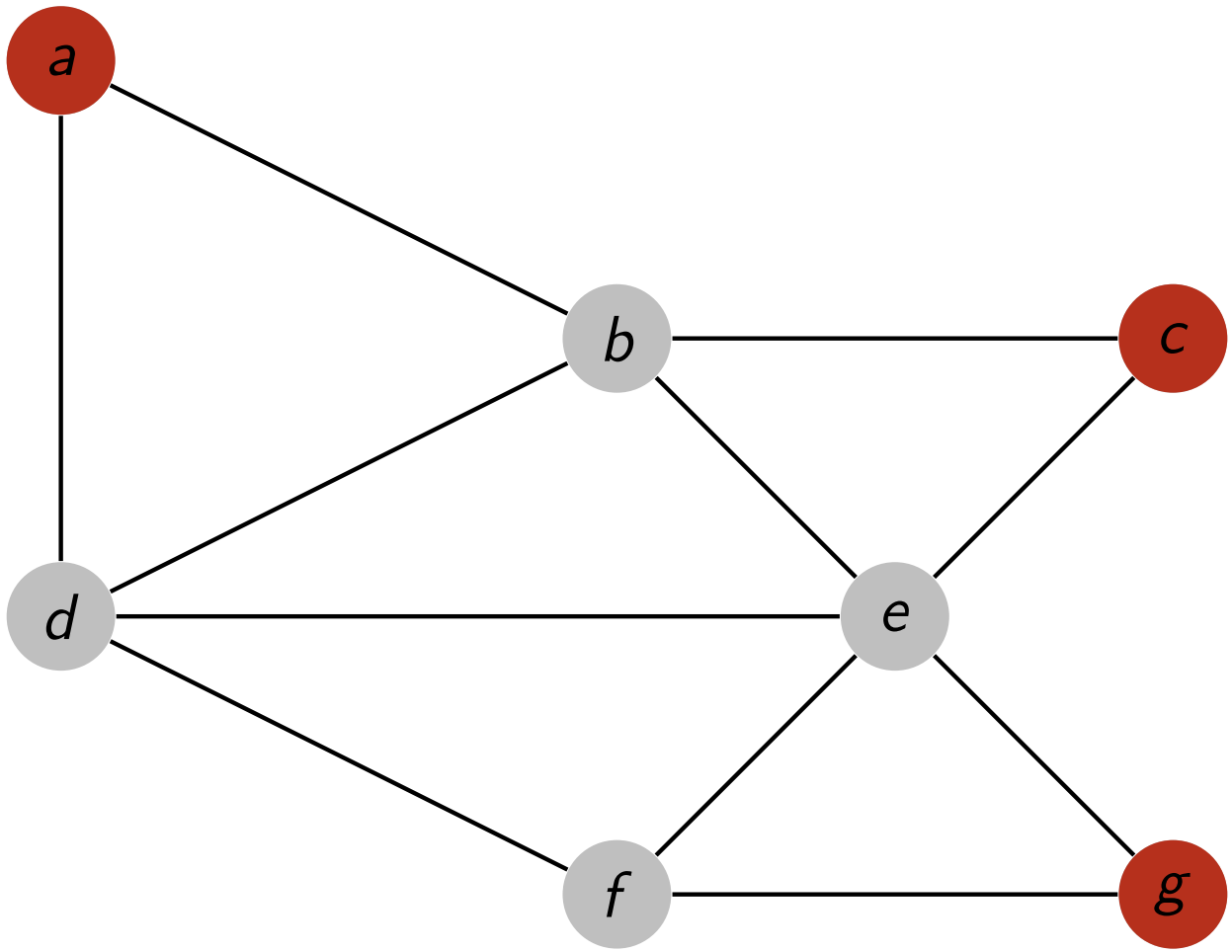
Ensemble indépendant

Ex : trouver un ensemble de sommets deux à deux non adjacents, de taille 3



Ensemble indépendant

Ex : trouver un ensemble de sommets deux à deux non adjacents, de taille 3



Ensemble indépendant

Problème : étant donné un **graphe**, un entier k, trouver un ensemble de sommets deux à deux non adjacents, de taille k

- **Entrée :** un graphe $G = (V, E)$ et $k > 0$ un entier
- **Sortie :** un ens. $W \subseteq V$ de sommets ind. tq $|W| = k$

Ensemble indépendant

Problème : étant donné un **graphe**, un entier k, trouver un ensemble de sommets deux à deux non adjacents, de taille k

- **Entrée :** un graphe $G = (V, E)$ et $k > 0$ un entier
- **Sortie :** un ens. $W \subseteq V$ de sommets ind. tq $|W| = k$

Problème de décision :

- **Entrée :** un graphe $G = (V, E)$ et $k > 0$ un entier
- **Question :** G admet-il un ens. indépendant de taille k ?

Ensemble indépendant

Problème : étant donné un **graphe**, un entier k, trouver un ensemble de sommets deux à deux non adjacents, de taille k

- **Entrée :** un graphe $G = (V, E)$ et $k > 0$ un entier
- **Sortie :** un ens. $W \subseteq V$ de sommets ind. tq $|W| = k$

Problème de décision :

- **Entrée :** un graphe $G = (V, E)$ et $k > 0$ un entier
- **Question :** G admet-il un ens. indépendant de taille k ?

Langage :

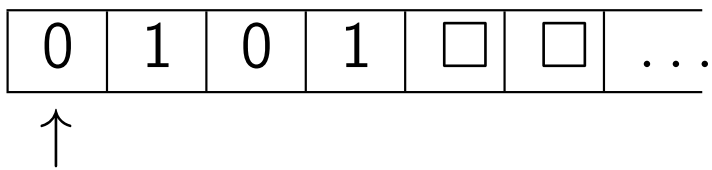
$$\text{IND-SET} = \{(G, k) : G \text{ admet un ens. ind. de taille } k\}$$

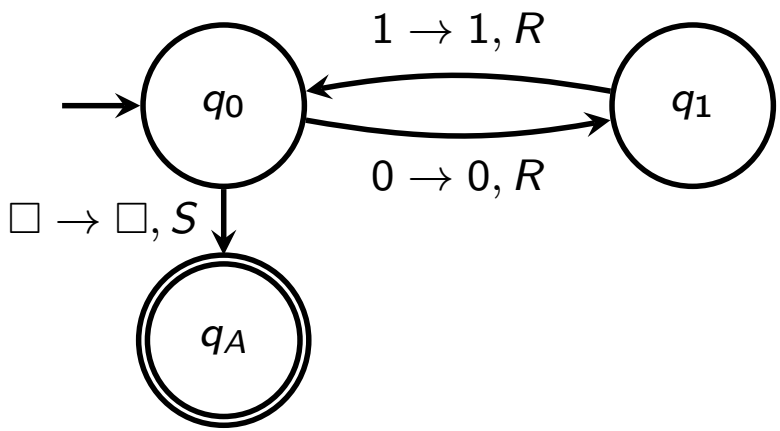
Machines de Turing

Introduction aux machines de Turing

- structure de **contrôle finie** (similaire à un automate)
- **mémoire infinie** (ruban ou bande) à accès **séquentiel**
- chaque transition **lit** un symbole, **écrit** un symbole, **déplace la tête de lecture** et change l'**état** de contrôle

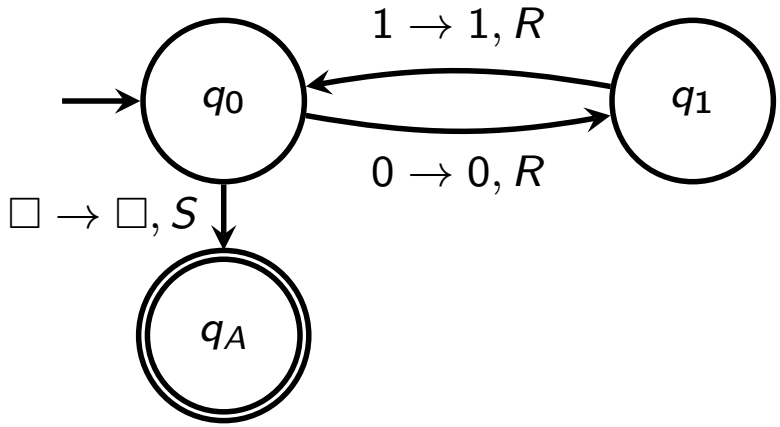
Ruban :





$$q_0, 0 \to q_1, 0, R$$
$$q_0, \square \to q_A, \square, S$$
$$q_1, 1 \to q_0, 1, R$$

Exécution d'une machine de Turing



$$q_0, 0 \to q_1, 0, R$$
$$q_0, \square \to q_A, \square, S$$
$$q_1, 1 \to q_0, 1, R$$

Démo : `https://turingmachinesimulator.com/` **sur** `01_slides.txt`

Entrée : 0101

$(q_0, \uparrow 0101) \to (q_1, 0 \uparrow 101) \to (q_0, 01 \uparrow 01) \to (q_1, 010 \uparrow 1) \to (q_0, 0101 \uparrow) \to$
$(q_A, 0101 \uparrow)$

Entrée : 011

$(q_0, \uparrow 011) \to (q_1, 0 \uparrow 11) \to (q_0, 01 \uparrow 1)$

Définition formelle

Une **machine de Turing** $M = (\Sigma, Q, q_{start}, F, \delta)$ **à** $k \geq 1$ **rubans** est constituée de :

- Σ est un alphabet fini contenant au moins $\square$ et $\triangleright$
- Q est un ensemble fini d'états, dont un **état initial** q_{start} et un ensemble d'**états acceptants** $F \subseteq Q$
- $\delta : Q \times \Sigma^k \to Q \times \Sigma^k \times \{L, R, S\}^k$ est une fonction de transition

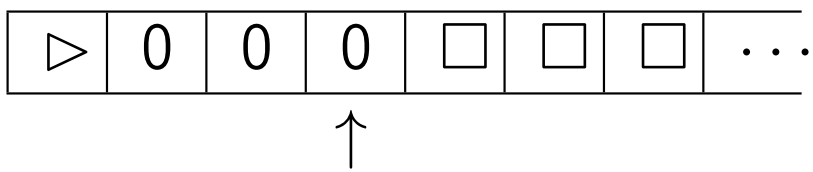
Alan Turing, "On Computable Numbers, with Application to the Entscheidungsproblem", Proc. London Math. Society, 45(2), pp. 230–265, 1936.

Ruban, tête de lecture

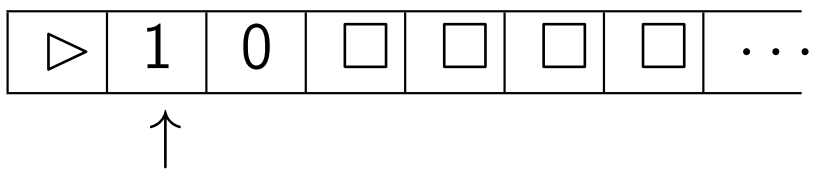
- Ruban : **infini** à droite, divisé en cases.
  - Chaque case contient **un symbole** de l'alphabet Σ
  - Le symbole "**blanc**" $\square$ est le symbole par défaut
  - La première case contient souvent un symbole choisi $\triangleright$
  - **Un ou plusieurs rubans**, en nombre constant

$\Sigma = \{0, 1, \triangleright, \square\}$

Ruban 1 :



Ruban 2 :

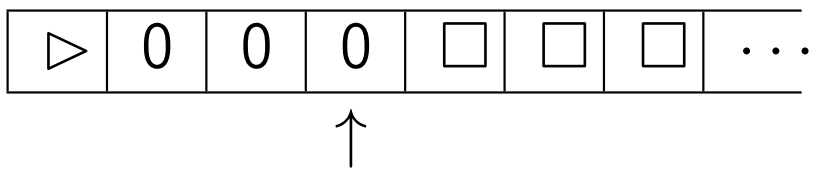


Ruban, tête de lecture

- Ruban : **infini** à droite, divisé en cases.
  - Chaque case contient **un symbole** de l'alphabet Σ
  - Le symbole "**blanc**" $\square$ est le symbole par défaut
  - La première case contient souvent un symbole choisi $\triangleright$
  - **Un ou plusieurs rubans**, en nombre constant
- Tête de lecture : une par ruban, sur une case.

$\Sigma = \{0, 1, \triangleright, \square\}$

Ruban 1 :



Ruban 2 :

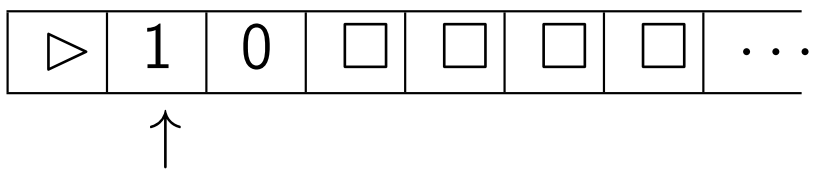
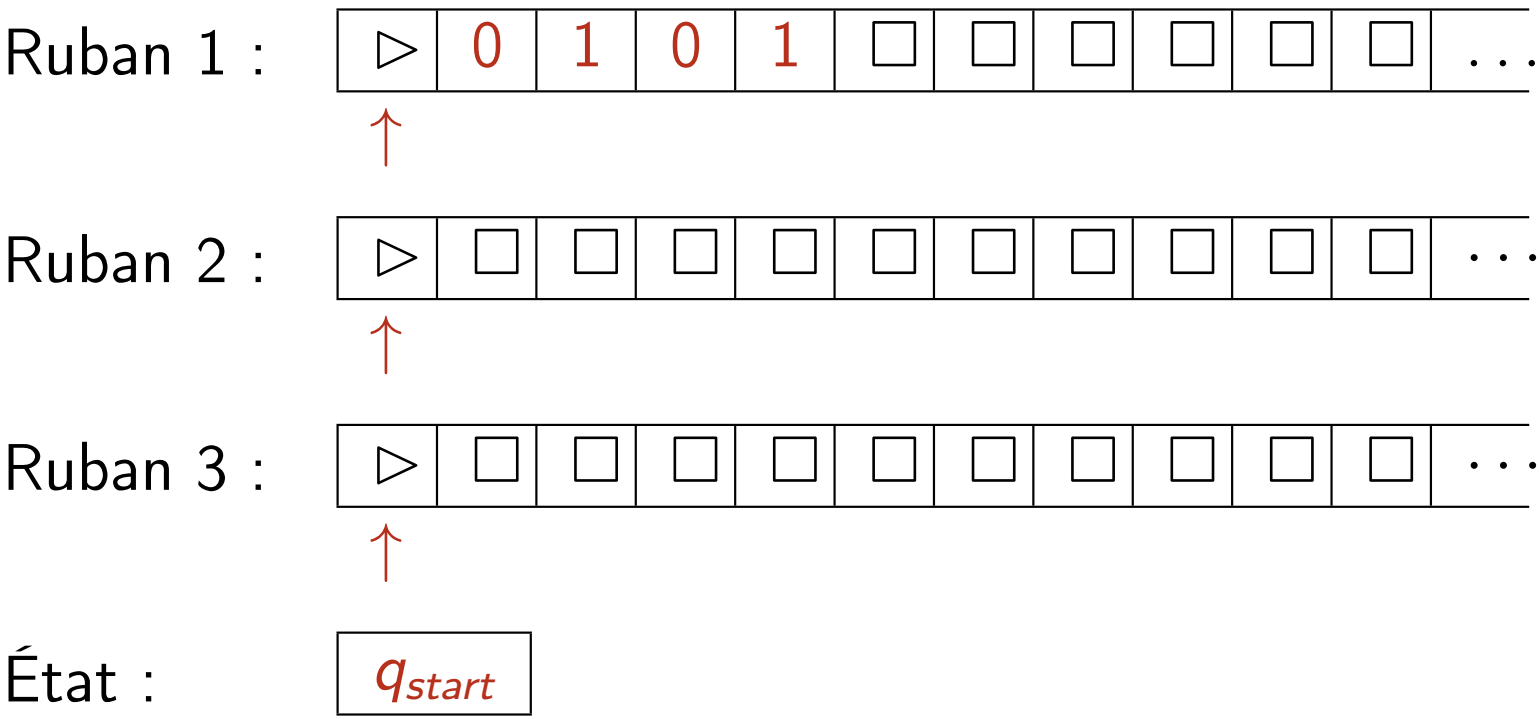


Illustration : configuration initiale

Mot d'entrée : 0101



Mathématiquement : $(q_{start}, \uparrow \triangleright 0101, \uparrow \triangleright, \uparrow \triangleright)$

Transition

En une transition ou étape ou pas de calcul :

- Le contenu des cases sur lesquelles sont positionnées les têtes de lectures est **lu** et (possiblement) **modifié**

Transition

En une transition ou étape ou pas de calcul :

- Le contenu des cases sur lesquelles sont positionnées les têtes de lectures est **lu** et (possiblement) **modifié**

- Chaque tête de lecture se **déplace d'une case** à gauche (L) si possible, à droite (R), ou elle reste à sa position (S)

Transition

En une transition ou étape ou pas de calcul :

- Le contenu des cases sur lesquelles sont positionnées les têtes de lectures est **lu** et (possiblement) **modifié**
- Chaque tête de lecture se **déplace d'une case** à gauche (L) si possible, à droite (R), ou elle reste à sa position (S)
- L'état de la machine est mis à jour

Transition

En une transition ou étape ou pas de calcul :

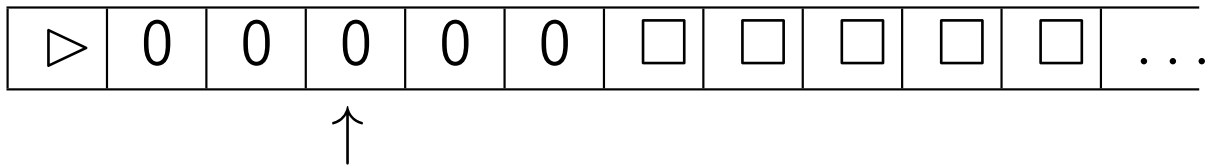
- Le contenu des cases sur lesquelles sont positionnées les têtes de lectures est **lu** et (possiblement) **modifié**
- Chaque tête de lecture se **déplace d'une case** à gauche (L) si possible, à droite (R), ou elle reste à sa position (S)
- L'état de la machine est mis à jour

Ces actions sont définies par une **fonction de transition** δ

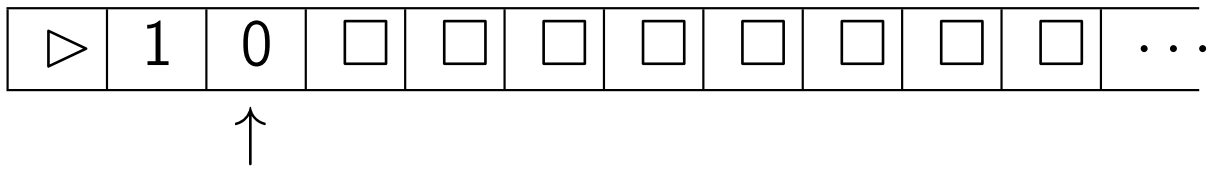
Illustration : transition

Exemple : $q_7, (0, 0, \square) \rightarrow q_{12}, (0, 1, 1), (R, S, S)$

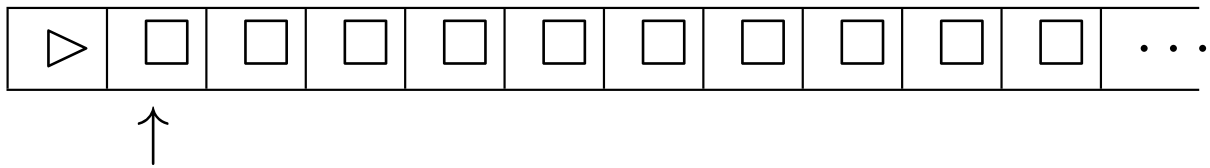
Ruban 1 :



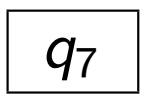
Ruban 2 :



Ruban 3 :



État :



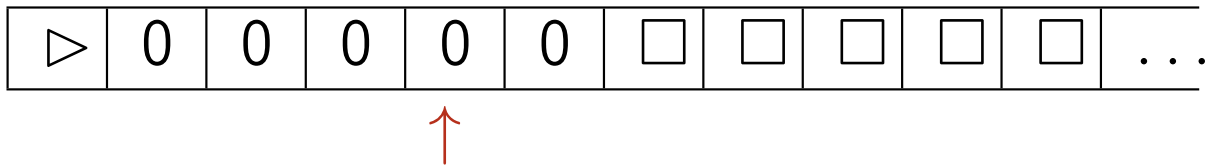
Mathématiquement :

$(q_7, \triangleright 00 \uparrow 000, \triangleright 1 \uparrow 0, \triangleright \uparrow) \vdash (q_{12}, \triangleright 000 \uparrow 00, \triangleright 1 \uparrow 1, \triangleright \uparrow 1)$

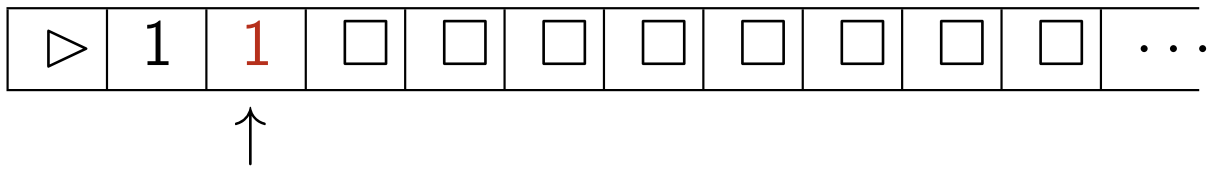
Illustration : transition

Exemple : $q_7, (0, 0, \square) \rightarrow q_{12}, (0, 1, 1), (R, S, S)$

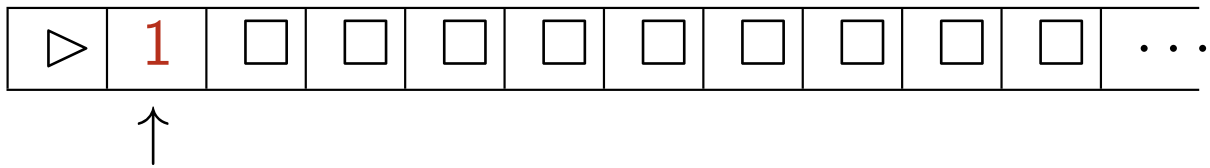
Ruban 1 :



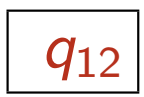
Ruban 2 :



Ruban 3 :



État :



Mathématiquement :

$(q_7, \triangleright 00 \uparrow 000, \triangleright 1 \uparrow 0, \triangleright \uparrow) \vdash (q_{12}, \triangleright 000 \uparrow 00, \triangleright 1 \uparrow 1, \triangleright \uparrow 1)$

Sémantique formelle : configuration

Soit $M = (\Sigma, Q, q_{start}, F, \delta)$ une MT à k rubans.

Configuration de M : tuple $(q, u_1 \uparrow v_1, \ldots, u_k \uparrow v_k)$ où :

- $q \in Q$ est l'état de contrôle
- $u_i v_i \in \Sigma^*$ est le contenu du ruban i
- la tête de lecture du ruban i est sur la première lettre de v_i

Configuration initiale : $(q_{start}, \uparrow \triangleright x, \uparrow \triangleright, \ldots, \uparrow \triangleright)$ où $x \in \Sigma^*$ est le **mot d'entrée**

Sémantique formelle : transition

La relation de transition $\vdash_M$ est définie par :

$$(q, u_1 \uparrow v_1, \ldots, u_k \uparrow v_k) \vdash_M (q', u'_1 \uparrow v'_1, \ldots, u'_k \uparrow v'_k)$$

s'il existe une transition :

$$q, (a_1, \ldots, a_k) \to q', (b_1, \ldots, b_k), (d_1, \ldots, b_k)$$

de δ telle que pour tout $i \in [1; k]$, $v_i = a_i x$ et :

- si $d_i = S$, alors $u'_i = u_i$ et $v'_i = b_i x$
- si $d_i = R$, alors $u'_i = u_i b$ et $v'_i = x$
- si $d_i = L$, alors $u_i = yc$ et $u'_i = y$ et $v'_i = cb_i x$

NB : pour $d_i = L$, on a u_i non vide (tête pas sur la 1ère case)

Sémantique formelle : exécution

Une exécution de M sur le mot d'entrée x est une suite **infinie**, ou **finie et maximale**, de configurations :

$$c_0 c_1 \ldots c_i \ldots$$

telle que c_0 est la configuration initiale, et pour tout $i \geq 0$, $c_i \vdash_M c_{i+1}$.

Le mot d'entrée x est :

- accepté si l'exécution de M sur x est finie et aboutit à une configuration finale $(q, u_1 \uparrow v_1, \ldots, u_k \uparrow v_k)$ avec $q \in F$;
- rejeté sinon : l'exécution est infinie, ou elle aboutit dans une configuration non acceptante.

Exemple : palindrome

Palindrome : mot qui se lit de la même façon de gauche à droite, et de droite à gauche

$\rightarrow$ formellement : w est un palindrome si $w = \mathit{mirroir}(w)$.

Exemples : *RADAR*, *LAVAL*, *ABBA*, ...

Palindromes binaires : ϵ, 0, 1, 00, 11, 000, 010, ...

$$\mathrm{PAL} = \{w \in \{0,1\}^* : w = \mathrm{mirroir}(w)\}$$

Décider PAL

Objectif : écrire une MT à 2 rubans qui, sur l'entrée x :

- s'arrête dans l'état $q_{yes} \in F$ si $x \in \text{PAL}$
- s'arrête dans l'état $q_{no} \notin F$ si $x \notin \text{PAL}$

Décider PAL

Objectif : écrire une MT à 2 rubans qui, sur l'entrée x :

- s'arrête dans l'état $q_{yes} \in F$ si $x \in \text{PAL}$
- s'arrête dans l'état $q_{no} \notin F$ si $x \notin \text{PAL}$

Principe :

Décider PAL

Objectif : écrire une MT à 2 rubans qui, sur l'entrée x :

- s'arrête dans l'état $q_{yes} \in F$ si $x \in \text{PAL}$
- s'arrête dans l'état $q_{no} \notin F$ si $x \notin \text{PAL}$

Principe :

1. Recopier le mot d'entrée du 1er ruban sur le 2ème ruban

Décider PAL

Objectif : écrire une MT à 2 rubans qui, sur l'entrée x :

- s'arrête dans l'état $q_{yes} \in F$ si $x \in \text{PAL}$
- s'arrête dans l'état $q_{no} \notin F$ si $x \notin \text{PAL}$

Principe :

1. Recopier le mot d'entrée du 1er ruban sur le 2ème ruban
2. Placer la tête de lecture du 1er ruban sur la première lettre, et celle du 2ème ruban sur la dernière lettre

Décider PAL

Objectif : écrire une MT à 2 rubans qui, sur l'entrée x :

- s'arrête dans l'état $q_{yes} \in F$ si $x \in \text{PAL}$
- s'arrête dans l'état $q_{no} \notin F$ si $x \notin \text{PAL}$

Principe :

1. Recopier le mot d'entrée du 1er ruban sur le 2ème ruban
2. Placer la tête de lecture du 1er ruban sur la première lettre, et celle du 2ème ruban sur la dernière lettre
3. Lire le 1er ruban de **gauche à droite** et le 2ème ruban de **droite à gauche** en **comparant lettre à lettre**

Décider PAL

Objectif : écrire une MT à 2 rubans qui, sur l'entrée x :

- s'arrête dans l'état $q_{yes} \in F$ si $x \in \text{PAL}$
- s'arrête dans l'état $q_{no} \notin F$ si $x \notin \text{PAL}$

Principe :

1. Recopier le mot d'entrée du 1er ruban sur le 2ème ruban
2. Placer la tête de lecture du 1er ruban sur la première lettre, et celle du 2ème ruban sur la dernière lettre
3. Lire le 1er ruban de **gauche à droite** et le 2ème ruban de **droite à gauche** en **comparant lettre à lettre**
4. Accepter si les lettres lues sont identiques, sinon rejeter

Machine M_{PAL}

$\Sigma = \{\square, \triangleright, 0, 1\}$
$Q = \{q_{start}, q_{copy}, q_{left}, q_{test}, q_{yes}, q_{no}\}$
$F = \{q_{yes}\}$

$\delta =$

q_{start}	$(\triangleright, \triangleright)$		q_{copy}	$(\triangleright, \triangleright)$	(R, R)
q_{copy}	$(a, \square)$		q_{copy}	(a, a)	(R, R)
q_{copy}	$(\square, \square)$		q_{left}	$(\square, \square)$	(L, S)
q_{left}	$(a, \square)$		q_{left}	$(a, \square)$	(L, S)
q_{left}	$(\triangleright, \square)$		q_{test}	$(\triangleright, \square)$	(R, L)
q_{test}	$(\square, \triangleright)$		q_{yes}	$(\square, \triangleright)$	(S, S)
q_{test}	$(a, b), a = b$		q_{test}	(a, b)	(R, L)
q_{test}	$(a, b), a \neq b$		q_{no}	(a, b)	(S, S)

où $a, b \in \{0, 1\}$.

Démo : `https://turingmachinesimulator.com/` **sur** `palindrome_slides.txt`

M_{PAL} en langage C

```
int palindrome(char s1[]) // input tape
{
  char s2[SOME_SIZE];     // working tape
  int i1 = 0, i2 = 0;     // tape heads

  // copy
  for (; s1[i1] != 0; ++i1, ++i2)
    s2[i2] = s1[i1];
  s2[i2--] = 0;

  // rewind (i1=0 is faster!)
  for (; i1 > 0; --i1);

  // check
  for (; i2 >= 0; ++i1, --i2)
    if (s1[i1] != s2[i2])
      return 0;           // REJECT
  return 1;               // ACCEPT
}
```

En C, meilleure solution **sans** `s2`, avec **deux "têtes"** `i1` et `i2`

Acceptation vs. Décision

Soit L un langage et M une MT.

M accepte L ssi pour tout mot w

- si $w \in L$, M s'arrête dans $q \in F$
- si $w \notin L$, M ne s'arrête pas, ou M s'arrête dans $q \notin F$

Acceptation vs. Décision

Soit L un langage et M une MT.

M accepte L ssi pour tout mot w

- si $w \in L$, M s'arrête dans $q \in F$
- si $w \notin L$, M ne s'arrête pas, ou M s'arrête dans $q \notin F$

M décide L ssi pour tout mot w

- M s'arrête et
- si $w \in L$, M s'arrête en $q \in F$
- si $w \notin L$, M s'arrête en $q \notin F$

Acceptation vs. Décision

Soit L un langage et M une MT.

M accepte L ssi pour tout mot w

- si $w \in L$, M s'arrête dans $q \in F$
- si $w \notin L$, M ne s'arrête pas, ou M s'arrête dans $q \notin F$

M décide L ssi pour tout mot w

- M s'arrête et
- si $w \in L$, M s'arrête en $q \in F$
- si $w \notin L$, M s'arrête en $q \notin F$

NB : M décide $L \implies M$ accepte L

Acceptation vs. Décision

Soit L un langage et M une MT.

M accepte L ssi pour tout mot w

- si $w \in L$, M s'arrête dans $q \in F$
- si $w \notin L$, M ne s'arrête pas, ou M s'arrête dans $q \notin F$

M décide L ssi pour tout mot w

- M s'arrête et
- si $w \in L$, M s'arrête en $q \in F$
- si $w \notin L$, M s'arrête en $q \notin F$

NB : M décide $L \implies M$ accepte L

Ex : M_{PAL} décide PAL

Reconnaissable vs. Décidable

Un langage L est :

- reconnaissable ou récursivement énumérable s'il existe une MT qui **accepte** L

Reconnaissable vs. Décidable

Un langage L est :

- reconnaissable ou récursivement énumérable s'il existe une MT qui **accepte** L
- décidable ou récursif s'il existe une MT qui **décide** L

Reconnaissable vs. Décidable

Un langage L est :

- reconnaissable ou récursivement énumérable s'il existe une MT qui **accepte** L
- décidable ou récursif s'il existe une MT qui **décide** L

NB : L décidable $\Longrightarrow$ L reconnaissable

Reconnaissable vs. Décidable

Un langage L est :

- reconnaissable ou récursivement énumérable s'il existe une MT qui **accepte** L
- décidable ou récursif s'il existe une MT qui **décide** L

NB : L décidable $\Longrightarrow$ L reconnaissable

- L **décidable :** il existe un algorithme pour décider si $w \in L$
- L **reconnaissable :** il existe un semi-algorithme pour décider si $w \in L$.

Reconnaissable vs. Décidable

Un langage L est :

- reconnaissable ou récursivement énumérable s'il existe une MT qui **accepte** L
- décidable ou récursif s'il existe une MT qui **décide** L

NB : L décidable $\Longrightarrow$ L reconnaissable

- L **décidable :** il existe un algorithme pour décider si $w \in L$
- L **reconnaissable :** il existe un semi-algorithme pour décider si $w \in L$.

Ex : PAL est décidable (cf. M_{PAL})

Calcul et complexité en temps

Temps de calcul d'une **fonction** $f : \{0,1\}^* \to \{0,1\}$ exprimée en fonction de la taille de l'entrée

Calcul et complexité en temps

Temps de calcul d'une **fonction** $f : \{0,1\}^* \to \{0,1\}$ exprimée en fonction de la taille de l'entrée

Définition

Soit $f : \{0,1\}^* \to \{0,1\}$ et $T : \mathbb{N} \to \mathbb{N}$ deux fonctions, et M une MT.

Calcul et complexité en temps

Temps de calcul d'une **fonction** $f : \{0,1\}^* \to \{0,1\}$ exprimée en fonction de la taille de l'entrée

Définition

Soit $f : \{0,1\}^* \to \{0,1\}$ et $T : \mathbb{N} \to \mathbb{N}$ deux fonctions, et M une MT.

- M calcule f ssi pour tout $x \in \{0,1\}^*$, M s'arrête sur l'entrée x et avec $f(x)$ sur l'un de ses rubans.

Calcul et complexité en temps

Temps de calcul d'une **fonction** $f : \{0,1\}^* \to \{0,1\}$ exprimée en fonction de la taille de l'entrée

Définition

Soit $f : \{0,1\}^* \to \{0,1\}$ et $T : \mathbb{N} \to \mathbb{N}$ deux fonctions, et M une MT.

- M calcule f ssi pour tout $x \in \{0,1\}^*$, M s'arrête sur l'entrée x et avec $f(x)$ sur l'un de ses rubans.
- M calcule f en temps T ssi M calcule f, et M s'arrête sur l'entrée x après au plus $T(|x|)$ transitions.

Calcul et complexité en temps

Temps de calcul d'une **fonction** $f : \{0,1\}^* \to \{0,1\}$ exprimée en fonction de la taille de l'entrée

Définition

Soit $f : \{0,1\}^* \to \{0,1\}$ et $T : \mathbb{N} \to \mathbb{N}$ deux fonctions, et M une MT.

- M calcule f ssi pour tout $x \in \{0,1\}^*$, M s'arrête sur l'entrée x et avec $f(x)$ sur l'un de ses rubans.
- M calcule f en temps T ssi M calcule f, et M s'arrête sur l'entrée x après au plus $T(|x|)$ transitions.

Ex : M_{PAL} décide PAL en temps $3(|x|+1)$.

Calcul et complexité en temps

Temps de calcul d'une **fonction** $f : \{0,1\}^* \to \{0,1\}$ exprimée en fonction de la taille de l'entrée

Définition

Soit $f : \{0,1\}^* \to \{0,1\}$ et $T : \mathbb{N} \to \mathbb{N}$ deux fonctions, et M une MT.

- M calcule f ssi pour tout $x \in \{0,1\}^*$, M s'arrête sur l'entrée x et avec $f(x)$ sur l'un de ses rubans.
- M calcule f en temps T ssi M calcule f, et M s'arrête sur l'entrée x après au plus $T(|x|)$ transitions.

Ex : M_{PAL} décide PAL en temps $3(|x|+1)$.

NB : Complexité **au pire**