UE 4I900
Master Informatique, UPMC

# COMPLEX

COMPLEXité, algorithmes randomisés et approchés

B. Escoffier, S. Kedad-Sidhoum, F. Pascual, L. Perret (responsable)

# Contenu de l'UE

**S 1-2 :** Introduction à la théorie du calcul et à la complexité des problèmes

**S 3 :** Algorithmes d'approximation

**S 4 :** Méthodes arborescentes

**S 5 :** Algorithmes faiblement exponentiels

**S 6-7 :** Algorithmes randomisés

**S 8 :** Classes de complexité probabilistes

**S 9 :** Protocoles *Zero-Knowledge*

**S 10 :** Fin du cours/révision

# Organisation et contrôle des connaissances

- Un cours (2h) par semaine
- 2h TD-2h TME, ou 4h TD par semaine
- Deux mini-projets pendant les TME:
- Projet 1: semaines 3-4 (soutenances S5),
- Projet 2: semaines 6-8 (soutenances S9)
- Site de l'UE :
  http://www-salsa.lip6.fr/~perret/Site/teaching.html
- **TD-TME : passer dans le groupe du vendredi si possible**

# Organisation et contrôle des connaissances

- Devoir 1 (rapport+soutenance projet 1): 15%
- Devoir 2 (rapport+soutenance projet 2) : 15%
- Examen réparti 1 (novembre): 35%
- Examen réparti 2 (janvier) : 35%

# Chapitre 1
# Introduction à la théorie du calcul

**a.** Y a-t-il des **problèmes** que l'on ne **peut pas** résoudre avec un ordinateur, ie pour lesquels il n'existe pas d'**algorithme** ???

*Problème ?*

*Algorithme ?*

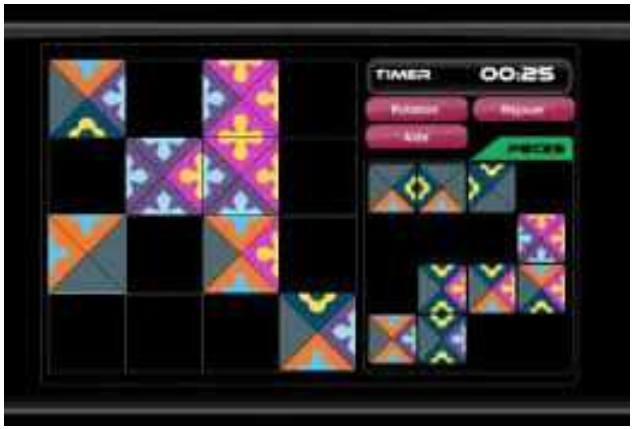
# Introduction à la théorie du calcul

**b.** Parmi les problèmes que l'on peut résoudre, y en a-t-il de plus difficiles que d'autres ?

GPS


Trajet optimal en quelques secondes

Eternity




Puzzle 16*16

$10^6$ €

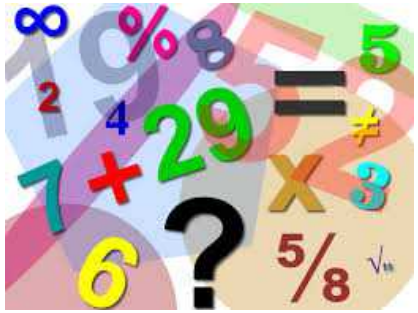
# Introduction à la théorie du calcul

## I. Problème, algorithme


Une donnée/

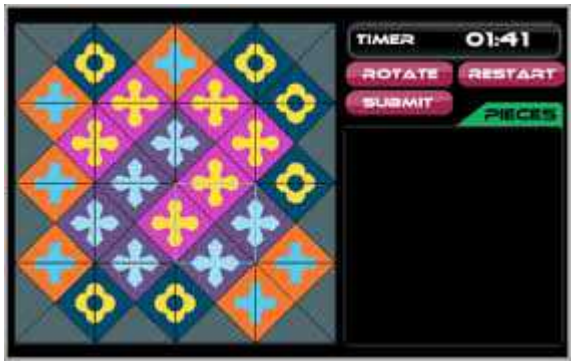
une instance

Une question



Une séquence d'instructions/

un algorithme

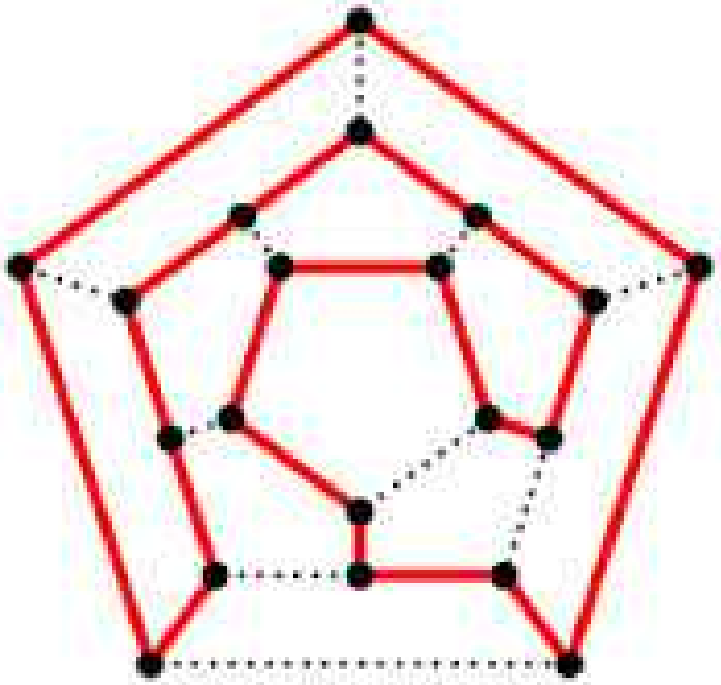



Une réponse/

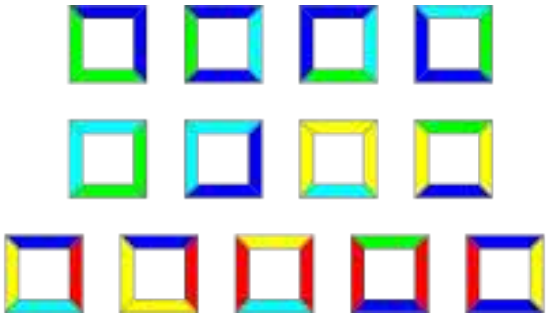
une solution

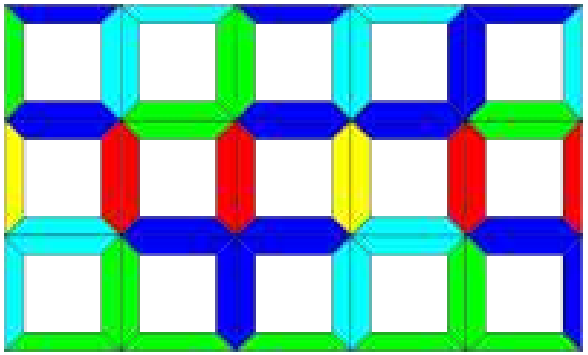
# Exemple 1: chaîne entre deux sommets

# Exemple 2: chaîne hamiltonienne



# Exemple 3: pavage du plan





...

...

# I. Problème, algorithme


Suite finie d'instructions.

But : donner la réponse à une question (résoudre un problème).

# I. Problème, algorithme

# I. Problème, algorithme

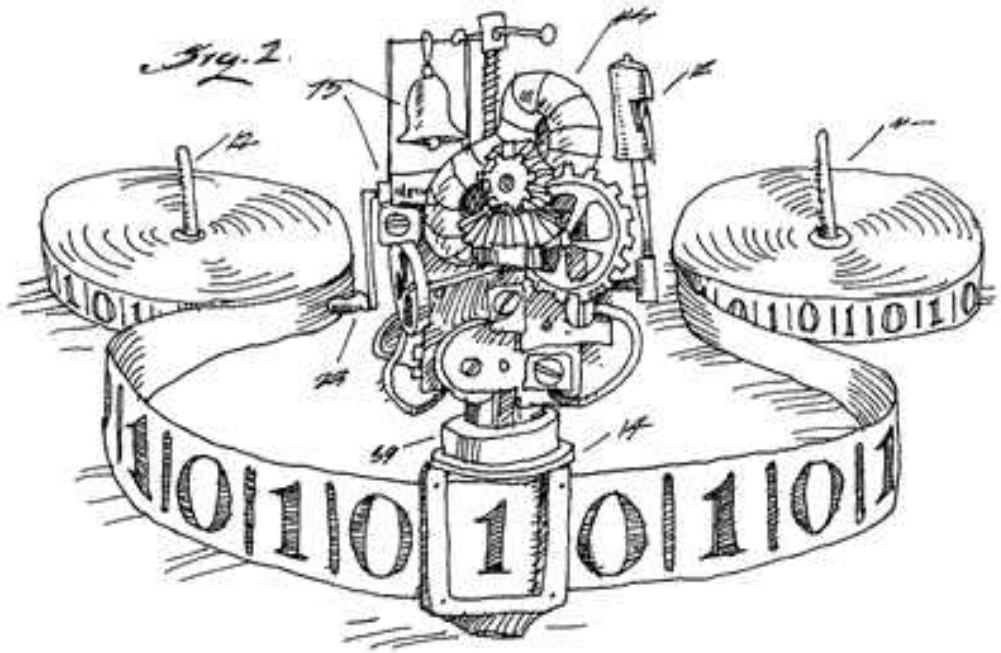
**Enoncé 2:** Peut-on prouver la cohérence de l'arithmétique? En d'autres termes, peut-on démontrer que les axiomes de l'arithmétique ne sont pas contradictoires ?

**Enoncé 10:** Trouver un algorithme déterminant si un polynôme à coefficients entiers a une racine entière.

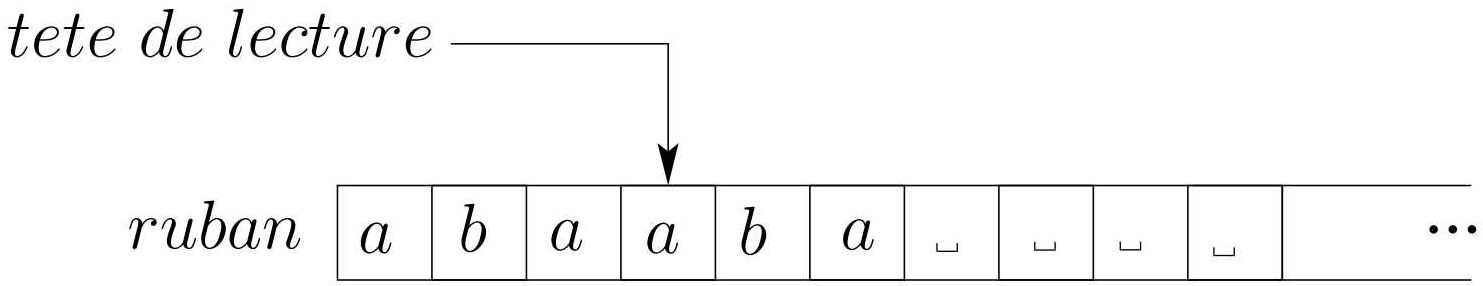
# II. Formalisation

# II. Formalisation

1. Problème et langage
2. Machine de Turing



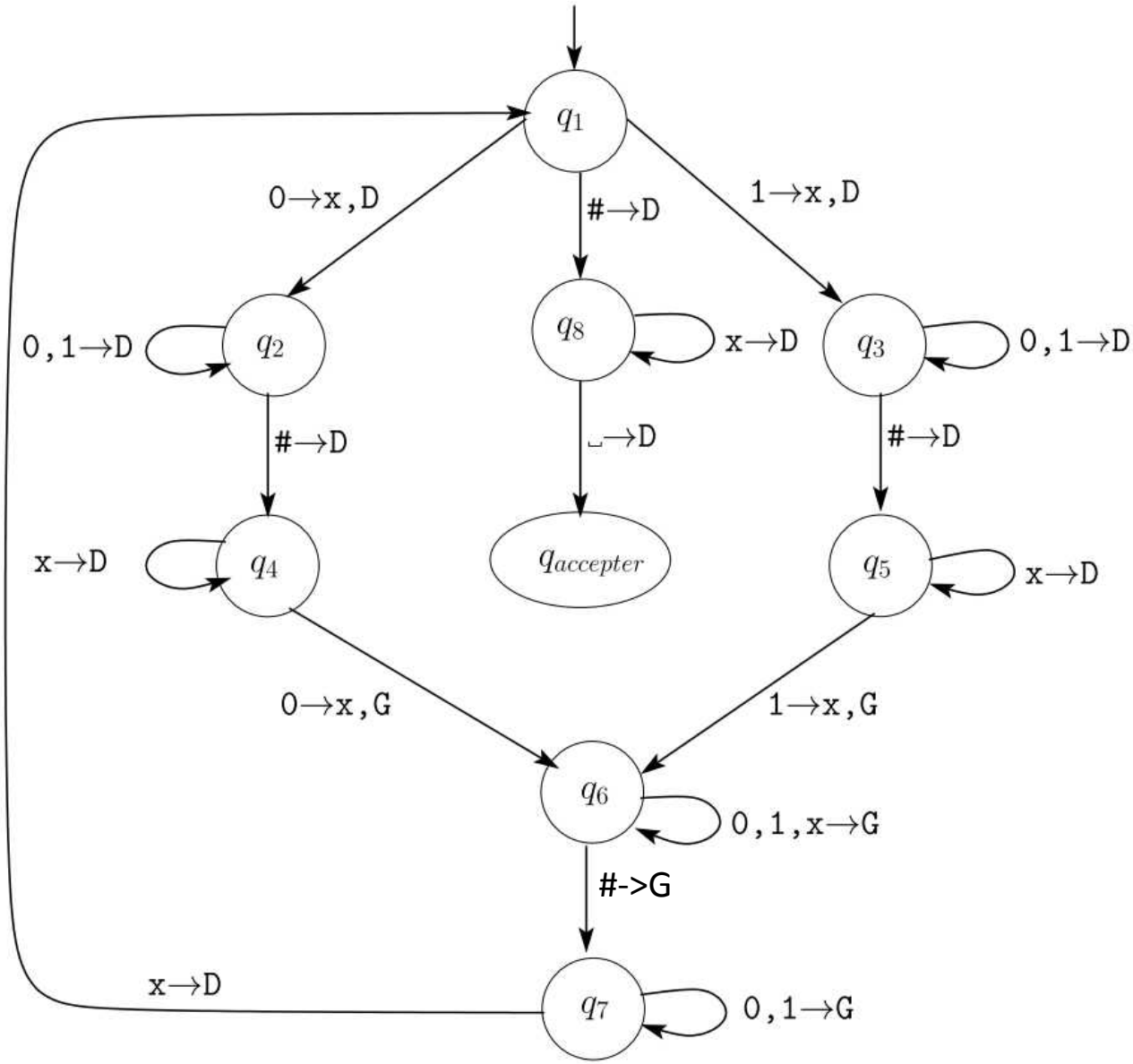
# II. Formalisation



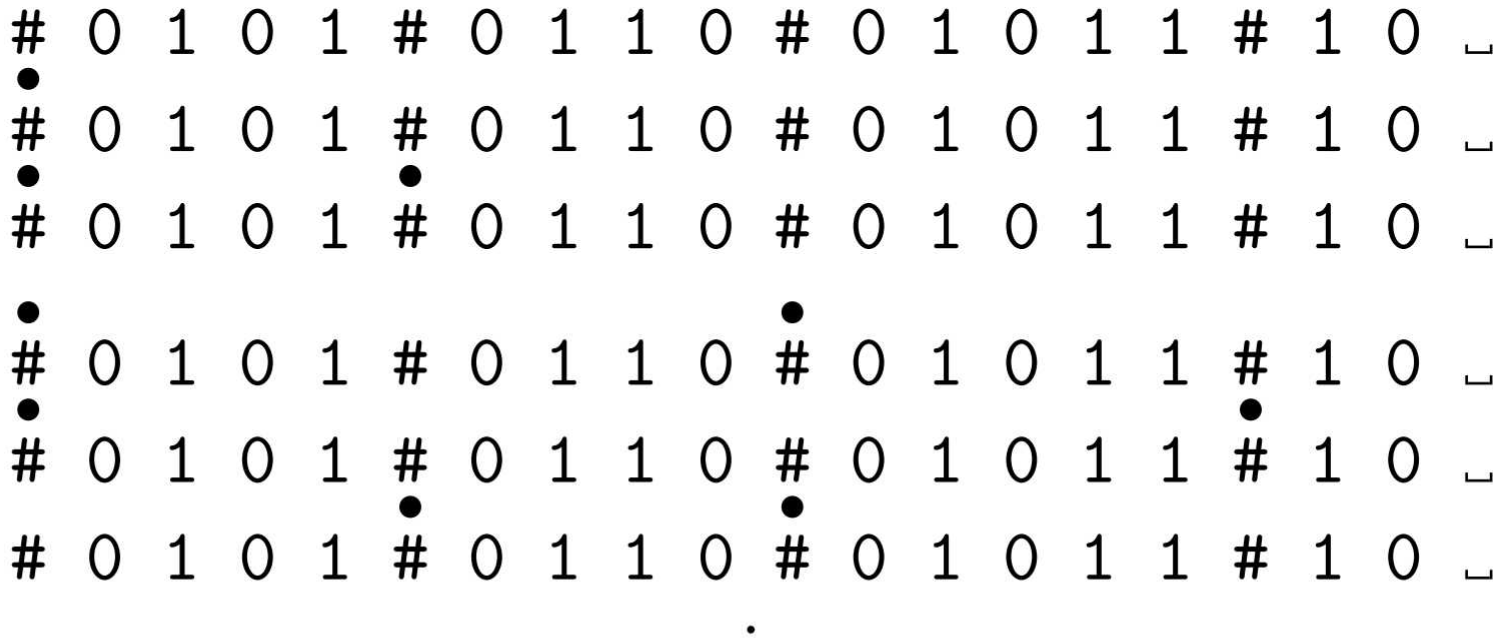
# II. Formalisation

Problème/langage : {w#w : w mot sur {0,1}}

```
0 0 1 0 1 1 1 0 1 0 1 # 0 0 1 0 1 1 1 0 1 0 1 ␣
x 0 1 0 1 1 1 0 1 0 1 # 0 0 1 0 1 1 1 0 1 0 1 ␣
x 0 1 0 1 1 1 0 1 0 1 # x 0 1 0 1 1 1 0 1 0 1 ␣
x x 1 0 1 1 1 0 1 0 1 # x 0 1 0 1 1 1 0 1 0 1 ␣
                      ⋮
x x x x x x x x x x x # x x x x x x x x x x x ␣
```

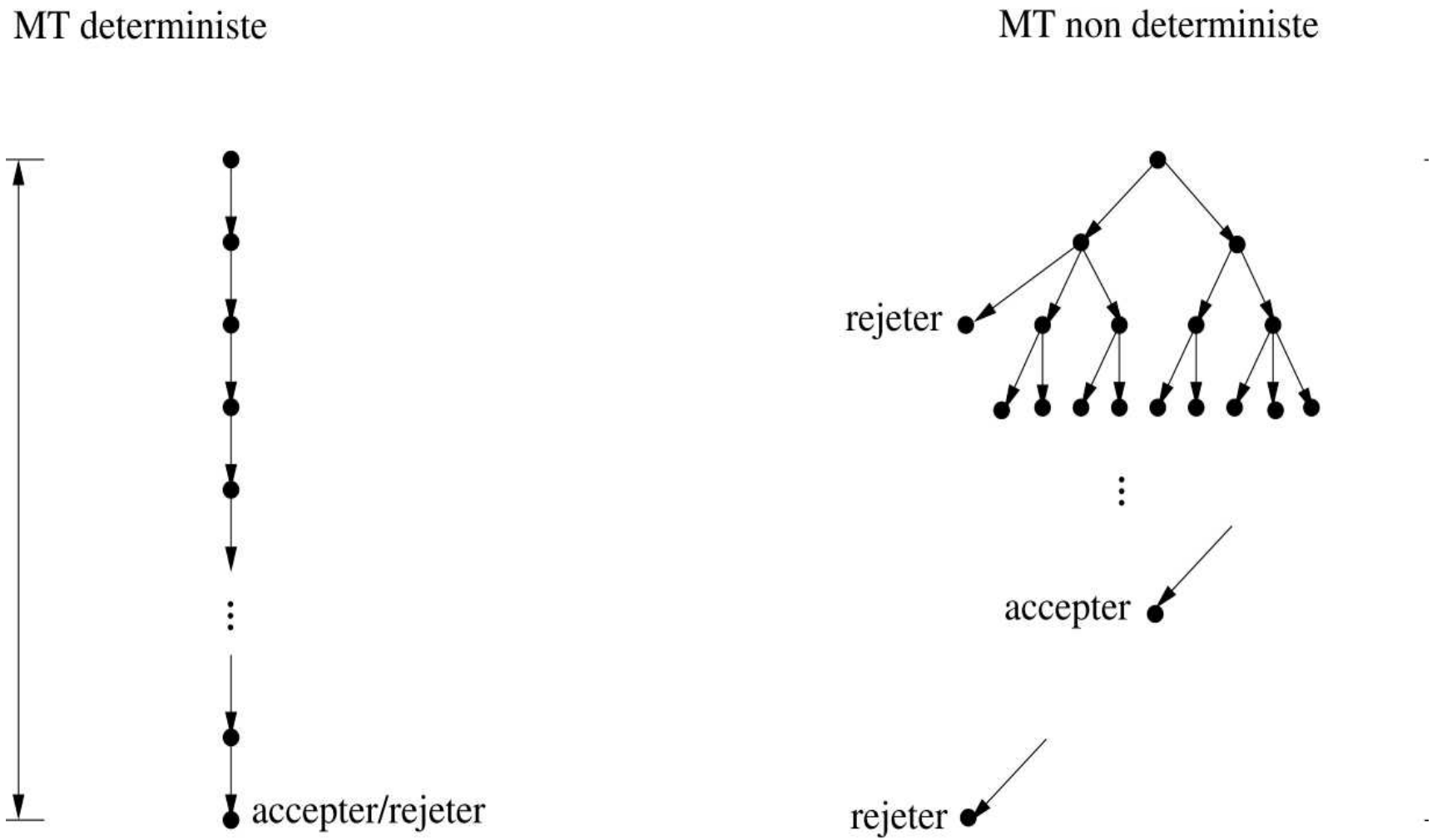

$q_1$
0→x,D
#→D
1→x,D
$q_2$
0,1→D
$q_8$
x→D
$q_3$
0,1→D
#→D
␣→D
#→D
$q_4$
x→D
$q_{accepter}$
$q_5$
x→D
0→x,G
1→x,G
$q_6$
0,1,x→G
#->G
$q_7$
x→D
0,1→G

# II. Formalisation



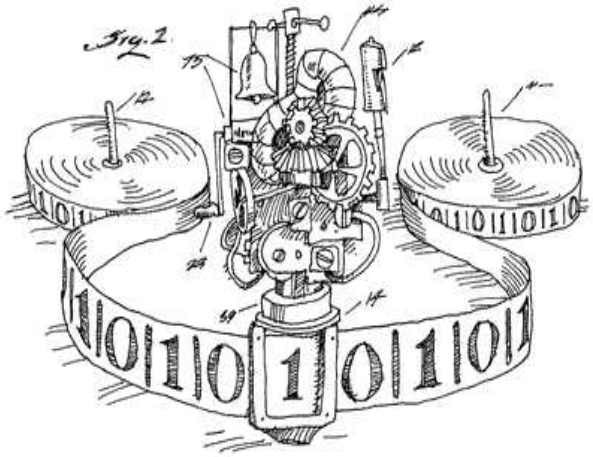

Problème/langage : éléments distincts

# 3. Machine de Turing non déterminisite



# Remarques pour conclure

- D'autres modèles ? Thèse de Church (Turing)

- 


??