

# Rudiments de Calculabilité et de Complexité

Paul Gastin

LSV, ENS Cachan, INRIA, CNRS, FRANCE

Informatique et sciences du numérique, 2 juin 2010

<http://www.lsv.ens-cachan.fr/~gastin/>



# Objectifs

## Calculabilité : Cerner les limites des machines ...

- ▶ Peut-on résoudre tous les problèmes avec un ordinateur ?
- ▶ Peut-on calculer toutes les fonctions avec un ordinateur ?
- ▶ Quels traitements des informations ?
- ▶ Pourra-t-on inventer un jour une machine qui pourra résoudre plus de problèmes que nos ordinateurs ?
- ▶ ...

## Complexité : ... et des temps de calculs

- ▶ Combien de temps faut-il pour résoudre un problème ?
- ▶ Trouvera-t-on un jour des algorithmes radicalement plus efficaces ?
- ▶ Est-ce que **facile à vérifier** équivaut à **facile à résoudre** ?
- ▶ ...

# Plan

## 1 Calculabilité

- Quelques exemples
- Pas de définition mais 2 outils
- Quel modèle de calcul ? Un peu d'histoire

## Complexité

## Bibliographie

# Sudoku

	2		6		8			
3	7					6		8
8		6		7				
4	9	1		5				6
		7				1		
2				4		3	7	5
				9		4		1
1		2					6	3
			5		3		2	

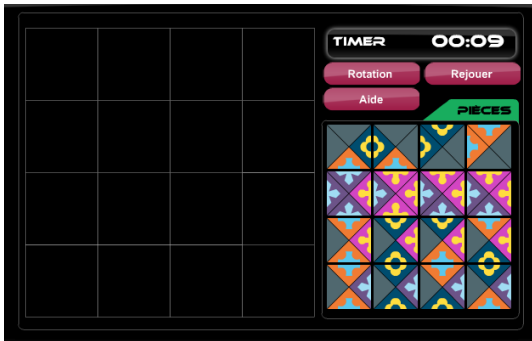
Peut-on résoudre automatiquement les problèmes de Sudoku ?

Codage : ? 2 ? 6 ? 8 ? ? ? 3 7 ? ...



# Eternity $16 = 4 \times 4$

<http://fr.eternityii.com/>

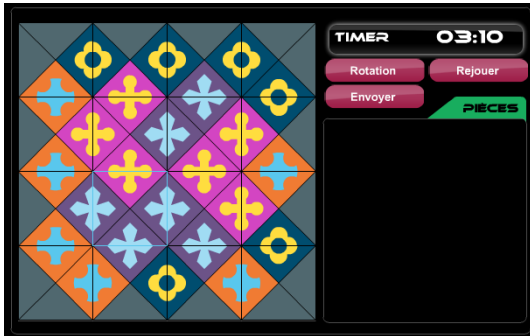


Peut-on résoudre automatiquement ce puzzle ?

Codage : GOBG BOGG BGGB OGGO  
VVRV VRRR ...

# Eternity $16 = 4 \times 4$

<http://fr.eternityii.com/>



Peut-on résoudre automatiquement ce puzzle ?

Codage : GOBG BOGG BGGB OGGO  
VVRV VRRR ...

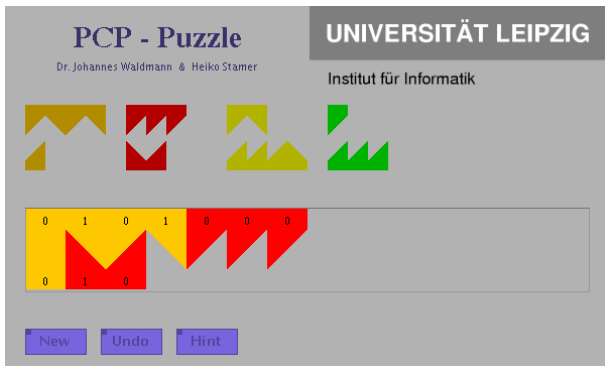






# Problème de Correspondance de Post

[http://www.theory.informatik.uni-kassel.de/~stamer/pcp/pcpcontest\\_en.html](http://www.theory.informatik.uni-kassel.de/~stamer/pcp/pcpcontest_en.html)



Peut-on résoudre le PCP automatiquement ?

Codage : (0101,0) (000,10) (01,0001) (0,000)

# Vérifier des programmes

Un problème des plus importants !

Codage : le texte source du programme.

Commençons simple :

- ▶ Mon programme va-t-il s'arrêter ?  
Sur toutes les données ?  
Sur au moins une donnée ?  
Sur cette donnée particulière ?
- ▶ Ma centrale risque-t-elle d'exploser ?

...

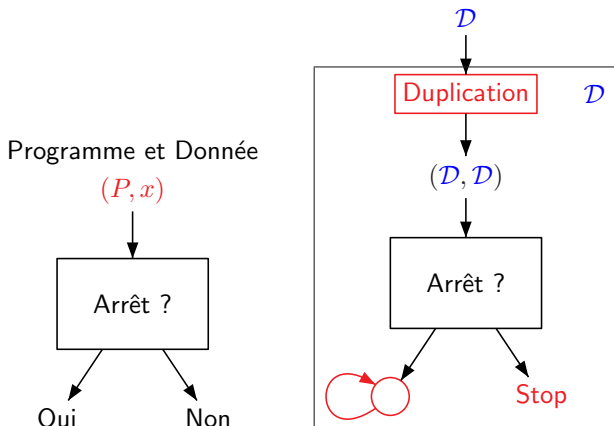
```
// Y a-t-il un risque de division par 0
```

```
z <- x / y
```

...



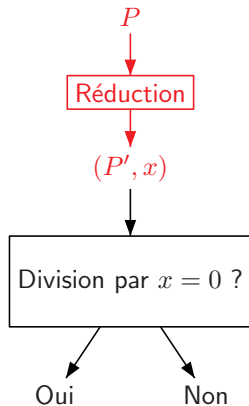
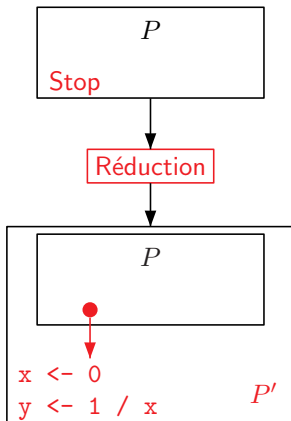
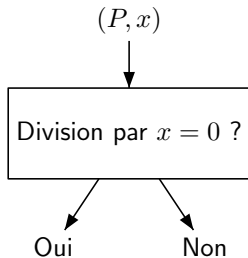
# Diagonalisation



Il n'y a pas de programme pour tester l'arrêt !

# Réduction

Programme et Variable



$P$  s'arrête si et seulement si  $P'$  divise par  $x = 0$

# Hilbert et les formalistes

David Hilbert, 1862 – 1943

Comment formaliser les preuves, par exemple sur les entiers naturels.

## Définir un système de preuves

### Des axiomes

$$x = x + 0$$

$$0 \neq x + 1$$

$$(x + y) + 1 = x + (y + 1)$$

### Des règles de déduction : modus ponens et généralisation

$$\frac{\varphi \quad \varphi \rightarrow \psi}{\psi}$$

$$\frac{\varphi}{\forall x, \varphi}$$

### Une preuve est une suite de déductions à partir des axiomes

## Propriétés d'un système de preuves :

**Cohérence** Si un énoncé est prouvable alors il est vrai.

On ne peut pas prouver un énoncé et aussi prouver son contraire.

**Complétude** Si un énoncé est vrai alors il est prouvable.

# Post et le calcul des propositions

Emil Post, 1897 – 1954

$$(x_1 \vee x_2) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_4) \wedge (\neg x_3 \vee \neg x_4)$$

## Théorème : 1921

Le calcul des propositions est cohérent et complet.

Mais il n'est pas très puissant et ne permet pas de formaliser des énoncés simples sur les entiers comme

Pour tout entier  $n$ , il existe un entier  $p \geq n$  qui est premier.

Pour cela on utilise le calcul des prédicats (logique du premier ordre).





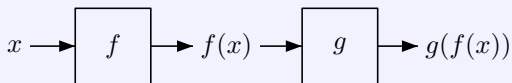
# Gödel et les fonctions calculables

## Les fonctions primitives récursives (1931 – 1934)

- ▶ L'addition (ou simplement l'incrémentation)

$$\begin{array}{r} 5 \ 8 \ 4 \ 5 \ 7 \\ + \quad 4 \ 2 \ 4 \ 5 \\ \hline 6 \ 2 \ 7 \ 0 \ 2 \end{array}$$

- ▶ La composition de fonctions



- ▶ L'itération de fonctions : par exemple la multiplication

$$\text{prod}(n, m) = \begin{cases} 0 & \text{si } m = 0 \\ n + \text{prod}(n, m - 1) & \text{sinon.} \end{cases}$$

`x <- 0; Pour i de 1 à m faire x <- x + n fpour; Retourner x`

Y a-t-il des fonctions calculables qui ne sont pas primitives récursives ?

Oui : la fonction d'Ackermann (Wilhelm, 1896 – 1962)

# Church et le $\lambda$ -calcul

Alonzo Church, 1903 – 1995

1936

- ▶ Church introduit le  $\lambda$ -calcul, ancêtre de la programmation fonctionnelle.
- ▶ Il montre que toutes les fonctions primitives récursives sont définissables dans le  $\lambda$ -calcul.

# Kleene et la minimisation

Stephen C. Kleene, 1909 – 1994

## Les fonctions $\mu$ -récursives (1936)

À la composition et à l'itération des fonctions,  
Kleene ajoute la définition par minimisation (non bornée):

$$g(x) = \min\{y \mid f(x, y) = 1 \text{ et } [0 \cdot \cdot y] \subseteq \text{dom}(f(x, -))\}$$

Remarque :  $\text{dom}(g) \subseteq \mathbb{N}$  mais la fonction  $g$  n'est pas forcément totale.

Calcul effectif de  $g(x)$  pour  $x \in \text{dom}(g)$ :

```
y <- 0; z <- f(x,y)
Tant que z ≠ 1 faire y <- y+1; z <- f(x,y) FinTantQue
Retourner y
```

## Théorème : Kleene 1936

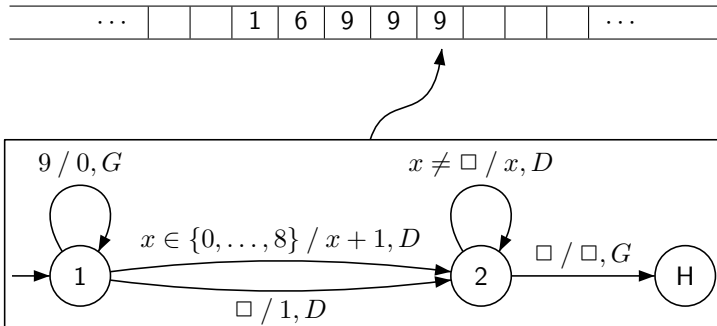
Une fonction est  $\mu$ -récursive si et seulement si elle est  $\lambda$ -définissable.

# Turing et les machines

Alan Turing, 1912 – 1954

<http://math.hws.edu/TMCM/java/xTuringMachine/>

<http://www.labri.fr/perso/betrema/MC/TuringJNLP.html>



## Théorème : Turing 1936

Une fonction est  $\mu$ -récursive si et seulement si elle est calculable par une machine de Turing.

# La thèse de Church

## Thèse de Church

La notion de **effectivement calculable** est capturée par les formalismes équivalents suivants :

- ▶ Les machines de Turing
- ▶ Les fonctions  $\mu$ -récursives
- ▶ Le  $\lambda$ -calcul
- ▶ Les processus combinatoires finis (Post, 1936)
- ▶ Les algorithmes de Markov (1954)
- ▶ Les machines RAM (Shepherdson and Sturgis, 1963)
- ▶ Les ordinateurs modernes

## Quelques portraits



David Hilbert  
1862 – 1943



Emil Post  
1897 – 1954



Kurt Gödel  
1906 – 1978



Alonzo Church  
1903 – 1995



Stephen C. Kleene  
1909 – 1994



Alan Turing  
1912 – 1954

# Plan

Calculabilité

2 Complexité

Bibliographie

# Objectifs

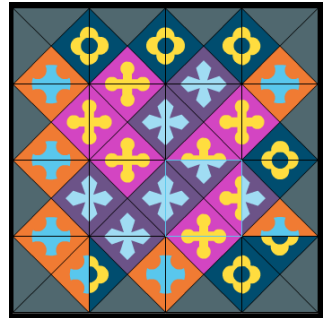
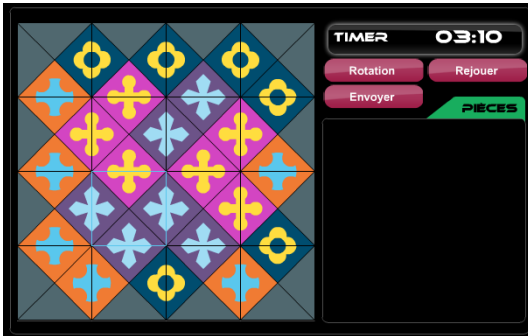
## Complexité des problèmes

- ▶ Combien de temps faut-il pour résoudre un problème ?
- ▶ Trouvera-t-on un jour des algorithmes radicalement plus efficaces ?
- ▶ Est-ce que **facile à vérifier** équivaut à **facile à résoudre** ?
- ▶ ...



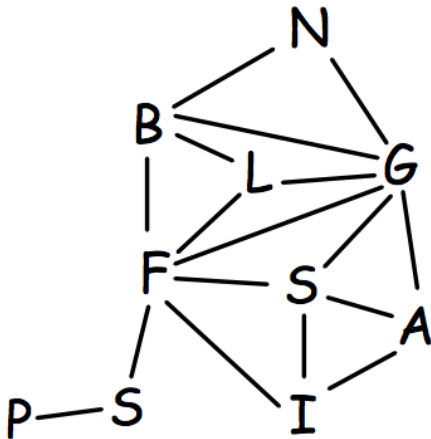
# Facile à vérifier : Eternity

<http://fr.eternityii.com/>



Facile = en temps polynomial

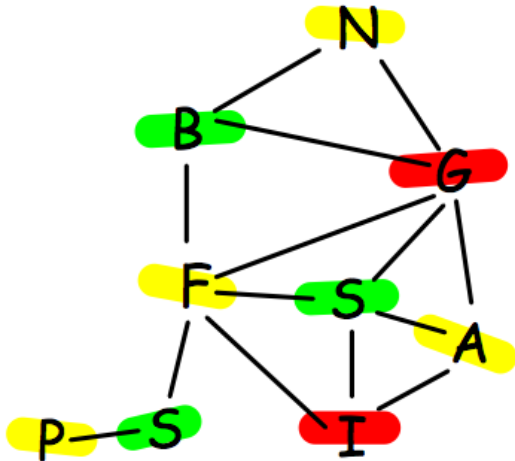
# Facile à trouver : 2-coloriage



Théorème :

Un graphe est 2-coloriable si et seulement si il n'a pas de cycle de longueur impaire.

# Facile à vérifier : 3-coloriage



Pourra-t-on trouver un jour un algorithme  
pour résoudre facilement le problème du 3-coloriage ?

# Facile à vérifier : somme d'entiers

Donnée : une suite d'entier  $n_1, n_2, \dots, n_k$  et un entier  $S$ .

Problème : Trouver une sous suite de somme  $S$ .

# Facile à vérifier : Le problème SAT

$$(x_1 \vee x_2) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_4) \wedge (\neg x_3 \vee \neg x_4)$$

$(x_1$  ou  $x_2)$  et  $(x_1$  ou non  $x_2$  ou  $x_3)$  et (non  $x_1$  ou  $x_2$  ou  $x_4)$  et (non  $x_3$  ou non  $x_4)$

$x_1$	Vrai
$x_2$	Faux
$x_3$	Faux
$x_4$	Vrai

Pourra-t-on trouver un jour un algorithme  
pour résoudre facilement la satisfaisabilité ?

# Réduction

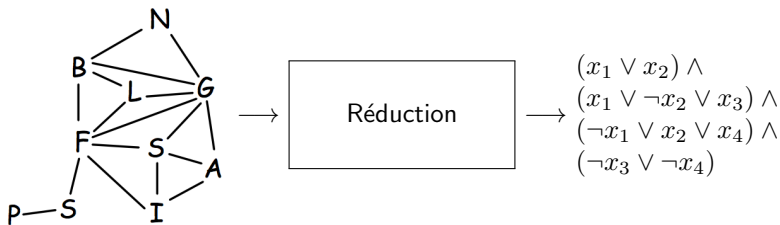
## Définition : Réduction

Une réduction d'un problème  $A$  à un problème  $B$  est un programme  $R$  qui **transforme très facilement** une donnée  $x$  de  $A$  en une donnée  $R(x)$  de  $B$  de telle sorte que

$x$  est solution de  $A$  si et seulement si  $R(x)$  est solution de  $B$

## Exemple :

Le problème du 3-coloriage se réduit au problème de satisfaisabilité.



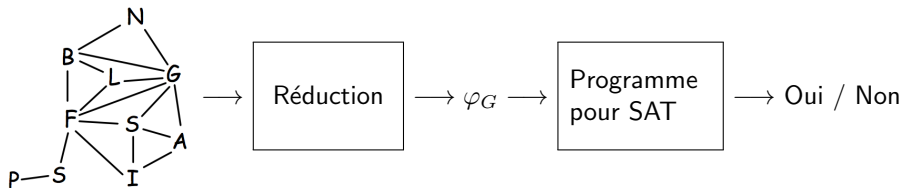
# Réduction

Proposition :

Si  $R$  est une réduction de  $A$  à  $B$  et que  $P$  est un programme pour résoudre  $B$  alors

$$R; P$$

est un programme pour résoudre  $A$ .



Si on trouve un algorithme polynomial pour résoudre SAT  
alors on aura un algorithme polynomial pour le 3-coloriage.

# Complexité

Remarque :

Si  $A$  se réduit à  $B$  et  $B$  se réduit à  $A$ , alors  $A$  et  $B$  ont la même complexité

Exemple :

Le problème du 3-coloriage est équivalent au problème de satisfaisabilité.



# Classes de complexité : P versus NP

## Définition : P et NP

- ▶ Un problème est dans la classe PTIME (P) si on peut le résoudre en temps polynomial.
- ▶ Un problème est dans la classe NPTIME (NP) si on peut deviner une réponse et vérifier en temps polynomial qu'elle est correcte.  
Formellement : non déterminisme.

Théorème : SAT est NP-complet    Stephen Cook, Leonid Levin, 1971

Le problème SAT est dans la classe NP et  
tout problème de la classe NP peut se réduire à SAT (NP-difficile),

Exemple : Les problèmes suivants sont NP-complets

3-coloriage, Voyageur de commerce, Somme d'entiers, Eternity, Sudoku, ...

La question à 1 million de dollars :  $P = NP$  ou  $P \neq NP$  ?

[http://www.claymath.org/millennium/P\\_vs\\_NP/](http://www.claymath.org/millennium/P_vs_NP/)

# Autres classes de complexité

## Différentes notions de “facile”

- ▶ Espace logarithmique
- ▶ Temps polynomial
- ▶ Espace polynomial
- ▶ Temps exponentiel

## Théorème : Hiérarchie

$\text{LOGSPACE} \subseteq \text{NLOGSPACE} \subseteq \text{PTIME} \subseteq \text{NPTIME} \subseteq \text{PSPACE} = \text{NPSPACE}$

$\text{PSPACE} \subseteq \text{EXPTIME} \subseteq \text{NEXPTIME} \subseteq \text{EXPSPACE} = \text{NEXPSPACE}$

# Plan

Calculabilité

Complexité

3 Bibliographie

# Autres classes de complexité

- [1] Jean-Michel Autebert.  
*Calculabilité et Décidabilité.*  
Masson, 1992.
- [2] Olivier Carton.  
*Langages formels, calculabilité et complexité.*  
Vuibert, 2008.
- [3] Richard L. Epstein, Walter A. Carnielli.  
*Computability: Computable Functions, Logic, and the Foundations of Mathematics.*  
Wadsworth, 2000 (2nd édition).
- [4] Dexter C. Kozen.  
*Automata and Computability.*  
Springer, 1997.
- [5] Christos Papadimitriou.  
*Computational complexity.*  
Addison-Wesley, 1995.