

3EL05

Chapitre 3 Complexité et mathématiques

Contenu

- Chapitre 1 : définition formelle
- Chapitre 2 : différentes façons d'approximer/d'estimer la complexité en pratique
- Chapitre 3 : étude de l'objet formel. 4 théorèmes sur la complexité (le 3e va vous surprendre)

C n'est pas calculable

- Par l'absurde, soit f calculant la complexité.

```
def more_complex(n):  
    i = 1  
    while True:  
        s = bin(i)[2:] # encodage binaire de i  
        if f(s) > n:  
            return s  
        i += 1
```

- `more_complex` termine toujours

C n'est pas calculable

```
def more_complex(n):  
    i = 1  
    while True:  
        s = bin(i)[2:] # encodage binaire de i  
        if f(s) > n:  
            return s  
        i += 1
```

`more_complex` termine donc peut générer s avec $n \rightarrow$

$$C(s) \leq C(\text{more_complex}) + C(n)$$

définition de s : $C(s) \geq n$

$\forall n \in \mathbb{N} \quad n \leq C(\text{more_complex}) + C(n)$ mais $C(n) \sim \log(n)$

C n'est pas calculable

...mais peut quand même servir pour définir
d'autres objets mathématiques et d'autres concepts

Probabilité algorithmique

- Quelle est la suite de 1223334444 ... ?
- $p(1\ 22\ 333\ 4444\ 55555)$?
- $p(1\ 22\ 333\ 4444\ 55055)$?
- Les programmes qui génèrent 55055 sont plus longs que ceux qui génèrent 55555

Probabilité algorithmique

- Quelle est la suite de 1223334444 ... ?
- $p(1\ 22\ 333\ 4444\ 55555)$?
- $p(1\ 22\ 333\ 4444\ 55055)$?
- Les programmes qui génèrent 55055 sont plus longs que ceux qui génèrent 55555
- $p(s) = 2^{-C(s)}$?

Tirage d'entiers

- Soit l'encodage binaire d'entiers
- Ça converge pas

Probabilité algorithmique

- $p_U(s) = \sum_{U(p)=s} 2^{-|p|}$?
- Problème : ça ne converge pas car on peut rallonger arbitrairement les programmes qui génèrent s
- Vous vous souvenez de Huffman vs Shannon ?
- $p_U(s) = \sum_{U(p)=s} 2^{-|p|}$ avec U machine de Turing préfixée

Programme normal → préfixé

- On double chaque bit sauf le dernier
- On indique le dernier bit par “01” ou “10”
 - 011100 → 001111110001
 - 0111 → 00111110
- Toute machine de Turing ordinaire peut être transformée en machine de Turing préfixée

Tirage d'entiers

- Soit l'encodage binaire préfixé d'entiers
- Ça converge

Probabilité algorithmique

- $p_U(s) = \sum_{U(p)=s} 2^{-|p|}$ avec U machine de Turing préfixée
- Pourquoi cette somme est bien définie ? / Pourquoi ça converge ?
- Souvenez-vous de l'arbre de Huffman
- Ou alors, les intervalles $[0.p, 0.p + 2^{-|p|}]$

Complexité préfixe

$$K_M(x) = \min_p \{l(p) \mid M(p) = x\}$$

avec M machine de Turing préfixée

A partir de maintenant:

- C = complexité ordinaire / *plain complexity*
- K = complexité préfixée / *prefix complexity*

⚠ Sur internet les gens écrivent l'un ou l'autre

Chain rule revisitée

Complexité ordinaire

Séparateur entre programmes

$$C(x) \leq C(y) + C(x|y) + O(\log(C(y)))$$



Longueur p_y

p_y

$p_{\{x|y\}}$

$$C(x, y) \leq C(y) + C(x|y) + O(\log(\min(C(x), C(y))))$$

Chain rule revisitée

$$K(x, y) = K(x) + K(y|x) + O(\log(K(x, y)))$$

$$K(x, y) = K(x) + K(y|p(x)) + O(1)$$



Plus court programme générant x

Complexité et aléa

- Qu'est-ce que l'aléa ? Qu'est-ce qu'un nombre aléatoire ?
- Richard von Mises (1919): *une séquence binaire est aléatoire si toute sous-séquence extraite indépendamment du contenu apparait 50% du temps (à la limite)*
Ca ne marche pas, cf la constante de Champernowne, déterministe mais a cette propriété

Complexité et aléa

- Qu'est-ce que l'aléa ? Qu'est-ce qu'un nombre aléatoire ?
- Emile Borel (1927) : soit une définition de l'aléatoire et soit n le plus petit nombre vérifiant cette définition $\rightarrow n$ n'est pas aléatoire
- Per Martin-Löf (1964) : cf TP

Complexité et aléa

- Qu'est-ce que l'aléa ? Qu'est-ce qu'un nombre aléatoire ?
- Kolmogorov (1965), Gregory Chaitin (1975) & Leonid Levin (1975) : une suite x est aléatoire si elle est peu compressible

$$\exists c, \forall m, K(x_m) > m - c$$



les m premiers
symboles de x

Complexité et aléa

$$\exists c, \forall m, K(x_m) > m - c$$

- π est compressible ?
- Avec la complexité, l'aléatoire et les probabilités deviennent des objets informatiques/computationnels (au lieu de mathématiques)

Théorème de Gödel

- Kurt Gödel (1931) : *certaines propositions mathématiques ne peuvent être démontrées*. La preuve de cette assertion est longue. Très longue.
- Gregory Chaitin (1970) : *il existe m_0 tel que $\forall m \geq m_0, \forall n$, l'assertion $K(n) > m$ est vraie mais n'est pas démontrable.*

Théorème de Gödel étendu

Soit S un système formel (par exemple l'arithmétique et ses axiomes de base)

S ne peut décrire des propositions de type $K(x) = n$ que si $K(S) + c > n$

“ S ne peut pas créer plus d'information qu'il n'en contient déjà”

Théorème de Gödel

Preuve par contraposée

Soit m tel que $\exists n, K(n) \geq m$ est démontrable

Je peux trouver n en énumérant tous les théorèmes
et en connaissant m : $K(n) \leq 2 \log(m) + O(1)$

$$m \leq K(n) \leq 2 \log(m) + O(1)$$

$$m \leq m_0$$

$\forall m, m > m_0 \Rightarrow \forall n, \boxed{K(n) \geq m}$ n'est pas démontrable

Ω de Chaitin

$$\Omega = \sum_{p: M(p) \text{ halts}} 2^{-l(p)}$$

- Ω n'est pas calculable
- Ω peut être approximé par une suite calculable, mais impossible de borner l'erreur

Soit S un système formel, tel que les propositions “ ω est un préfixe de Ω ” soit vraient. Elle ne peut être prouvée que si $l(\omega) \leq K(S) + c$

Classes algorithmiques dans \mathbb{N}

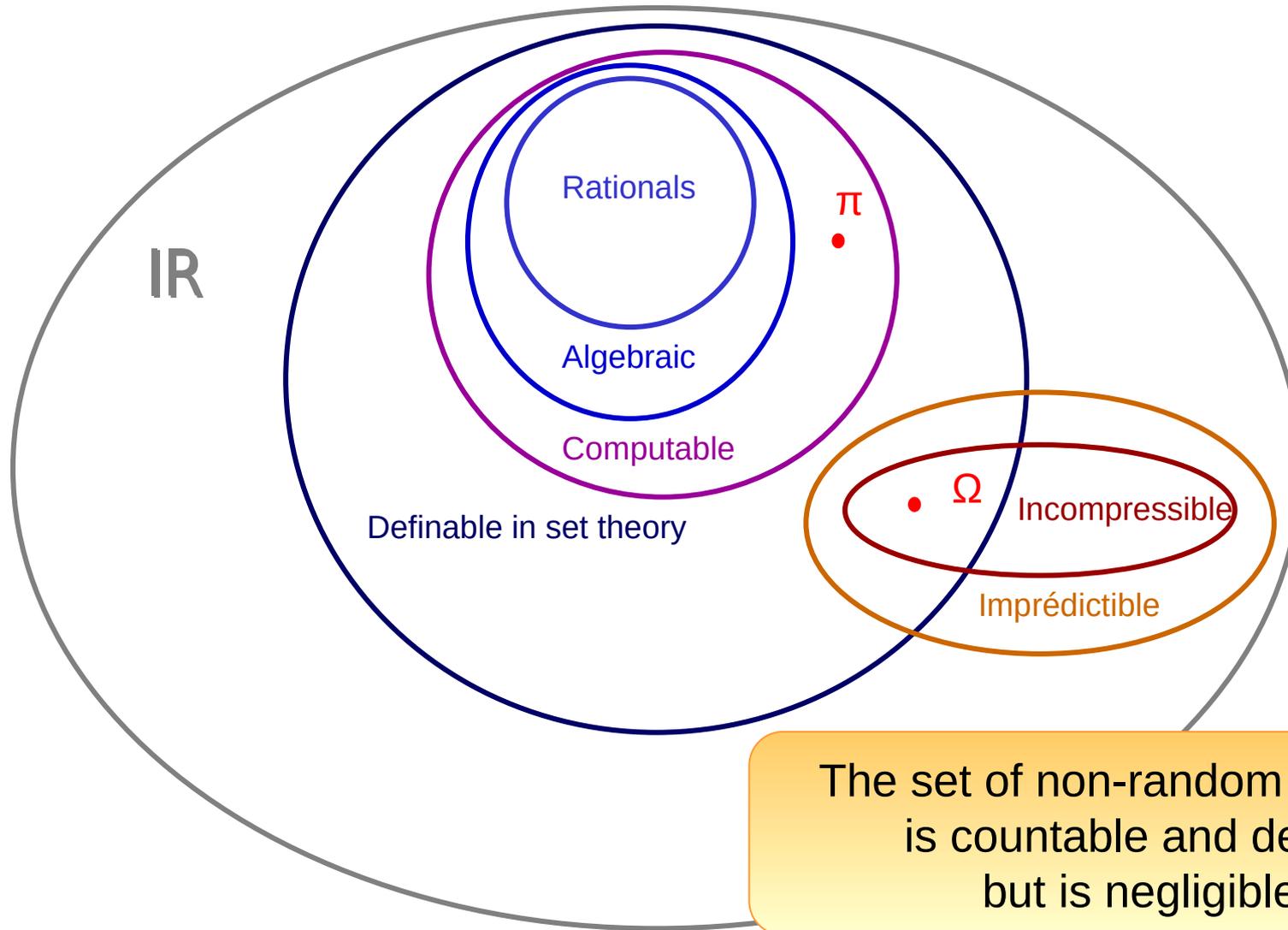
- Ensemble récursif ou décidable
 - *Définition*: Il existe une machine de Turing permettant de déterminer en temps fini si x est dans A
 - *Exemples*: Les nombres premiers
L'ensemble des solutions de $n+m=p$
- Ensemble récursivement énumérable
 - *Def*: Il existe un algorithme qui s'arrête si et seulement si l'entrée est dans A
 - *Ex*: Les théorèmes de l'arithmétique de Peano

Classes algorithmiques dans N

- Ensemble approximable
 - *Def*: Fini ou contient un sous-ensemble r.e.
 - *Ex*: L'ensemble des formules arithmétiques vraies
Contre-exemple : les formules vraies " $C(s) = n$ " (s séquence binaire finie)
- Ensemble incompressible
 - *Def*: Soit S un système formel correct pour l'appartenance (*correct* mais pas forcément *complet*). S peut générer au plus $K(S) + c$ membres de l'ensemble.
 - *Ex*: Les chiffres de Ω .

Classes algorithmiques dans \mathbb{R}

- Nombres rationnels
- Nombres algébriques
- Nombres transcendants
- Nombres normaux (fréquence limite b^{-k})
- Impredictibles (pas possible de gagner en pariant sur le prochain chiffre)
- Calculables
- Aléatoires (\rightarrow incompressibles)



The set of non-random numbers is countable and dense, but is negligible!

Conclusion

- La complexité ne peut pas être calculée
- Mais elle permet de prouver ou de définir:
 - L'aléatoire
 - Les probabilités
 - Le théorème de Gödel