

## Cours 4.1: Temps non-déterministe.

Olivier Bournez

bournez@lix.polytechnique.fr  
LIX, Ecole Polytechnique

INF561

Algorithmes et Complexité

1

## Au menu

Temps non-déterministe

Problèmes complets

Quelques résultats

2

## Sous menu

Temps non-déterministe

Classe NP

Algorithme non-déterministe

3

## Classe NP

- Soit  $M$  un alphabet fini.
- Un problème  $L \subset M^*$  est dans NP s'il existe un polynôme  $p: \mathbb{N} \rightarrow \mathbb{N}$  et un problème  $A$  (appelé *vérificateur pour L*) tel que
  - ▶ pour tout mot  $w$ ,  
 $w \in L$  si et seulement si  $\exists u \in M^*$  tel que  $\langle w, u \rangle \in A$ ,
  - ▶ et tel que déterminer si  $\langle w, u \rangle \in A$  admette un algorithme en temps  $p(|w|)$ .
- $u$  est appelé *un certificat* pour l'entrée  $w$ .

4

## Classe NP : autre écriture

- Soit  $M$  un alphabet fini.
- Un problème  $L \subset M^*$  est dans NP s'il existe un polynôme  $p: \mathbb{N} \rightarrow \mathbb{N}$  et un problème  $A$  dans P (appelé *vérificateur pour L*) tel que pour tout mot  $w$ ,

$$w \in L$$

si et seulement si

$$\exists u \in M^*, \text{ tel que } |u| \leq p(|w|) \text{ et } \langle w, u \rangle \in A.$$

- $u$  est appelé *un certificat* pour l'entrée  $w$ .

5

## Preuve

Les deux définitions sont équivalentes :

- en effet, être polynomial en  $\langle w, u \rangle$  implique être polynomial en  $|w|$  d'une part,
- et d'autre part, en temps polynomial en  $|w|$  on ne peut lire qu'un nombre polynomial de lettres de  $u$ , et donc on peut supposer le certificat réduit aux lettres que l'on lit, c'est-à-dire en un certificat de longueur polynomiale.

6

- Un problème  $L \subset M^*$  est dans  $\text{NP}_{\text{DT}}$  s'il existe un polynôme  $p: \mathbb{N} \rightarrow \mathbb{N}$  et un problème  $A$  dans  $\text{P}_{\text{DT}}$  tel que pour tout mot  $w$ ,

$$w \in L$$

si et seulement si

$$\exists u \in M^*, \text{ tel que } |u| \leq p(|w|) \text{ et } \langle w, u \rangle \in A.$$

- Un problème  $L \subset M^*$  est dans  $\text{NDP}_{\text{DT}}$  s'il existe un polynôme  $p: \mathbb{N} \rightarrow \mathbb{N}$  et un problème  $A$  dans  $\text{P}_{\text{DT}}$  tel que pour tout mot  $w$ ,

$$w \in L$$

si et seulement si

$$\exists u \in \{0,1\}^*, \text{ tel que } |u| \leq p(|w|) \text{ et } \langle w, u \rangle \in A.$$

7

## Sous menu

Temps non-déterministe

Classe NP

Algorithme non-déterministe

8

## Algorithme déterministe

Soit  $\Sigma$  une signature. Une règle sur  $\Sigma$  est soit :

- une *règle de mise à jour* de la forme

$f(t_1, \dots, t_k) \leftarrow t_0;$

- une *règle parallèle* de la forme

**par**  $R_1 R_2 \dots R_k$  **endpar** ;

- une règle de la forme

**if**  $\phi$  **then**  $R_1$ .

où  $f$  est un symbole  $k$ -aire de la signature  $\Sigma$ , et  $t_0, \dots, t_k$  sont des termes sur la signature  $\Sigma$ ,  $R_1, \dots, R_k$  sont des règles,  $\phi$  est un terme booléen.

9

## Algorithme non-déterministe

Soit  $\Sigma$  une signature. Une règle sur  $\Sigma$  est soit :

- une *règle de mise à jour* de la forme

$f(t_1, \dots, t_k) \leftarrow t_0;$

- une *règle parallèle* de la forme

**par**  $R_1 R_2 \dots R_k$  **endpar** ;

- une règle de la forme

**if**  $\phi$  **then**  $R_1$ .

- une règle de la forme

**guess**  $x$

où  $f$  est un symbole  $k$ -aire de la signature  $\Sigma$ , et  $t_0, \dots, t_k$  sont des termes sur la signature  $\Sigma$ ,  $R_1, \dots, R_k$  sont des règles,  $\phi$  est un terme booléen,  $x$  est un symbole de constante.

10

## Algorithme non-déterministe digital

Soit  $\Sigma$  une signature. Une règle sur  $\Sigma$  est soit :

- une *règle de mise à jour* de la forme

$f(t_1, \dots, t_k) \leftarrow t_0;$

- une *règle parallèle* de la forme

**par**  $R_1 R_2 \dots R_k$  **endpar** ;

- une règle de la forme

**if**  $\phi$  **then**  $R_1$ .

- une règle de la forme

**guess**  $x \in \{0, 1\}$

où  $f$  est un symbole  $k$ -aire de la signature  $\Sigma$ , et  $t_0, \dots, t_k$  sont des termes sur la signature  $\Sigma$ ,  $R_1, \dots, R_k$  sont des règles,  $\phi$  est un terme booléen,  $x$  est un symbole de constante.

11

## Sémantique

- chaque règle **guess** produit une interaction avec l'environnement :

▶ l'environnement choisi

- un  $m \in M$  dans le premier cas
- un  $m \in \{0, 1\}$  dans le second cas

▶ la règle a alors pour effet  $(x, \emptyset) \leftarrow m$ , c'est-à-dire que l'on affecte au contenu de l'emplacement  $x$  la valeur de  $m$ .

- À un algorithme n'est plus associé un système de transition, mais un système de transition non-déterministe.
- Une entrée  $w$  est acceptée en temps  $t(n)$  s'il existe un chemin de longueur  $t(n)$  entre un état initial et un état accepteur dans ce système de transition, où  $n$  est la longueur de l'entrée.

12

## Théorème

- Un problème  $L \subset M^*$  est dans NP si et seulement s'il est décidé en temps non-déterministe polynomial.
- Soit  $\mathfrak{M} = (M, f_1, \dots, f_u, r_1, \dots, r_v)$  une structure.  
Un problème  $L \subset M^*$  est dans  $\text{NP}_{\mathfrak{M}}$  si et seulement s'il est décidé en temps non-déterministe polynomial à paramètres sur  $\mathfrak{M}$ .
- Un problème  $L \subset M^*$  est dans  $\text{NDP}_{\mathfrak{M}}$  si et seulement s'il est décidé en temps non-déterministe digital polynomial à paramètres sur  $\mathfrak{M}$ .

13

## Au menu

Temps non-déterministe

Problèmes complets

Quelques résultats

14

## Sous menu

### Problèmes complets

#### Notion de réduction

NP-complétude

Existence de problèmes NP-complet

Premier problème NP-complet naturel

Autres problèmes NP-complets

Sur les réels

15

## Réduction

- Soient  $A$  et  $B$  deux problèmes d'alphabet respectifs  $M_A$  et  $M_B$ , et de langages respectifs  $L_A$  et  $L_B$ .
- Une *réduction de  $A$  vers  $B$*  est une fonction  $f : M_A^* \rightarrow M_B^*$  **calculable en temps polynomial** telle que
$$w \in L_A \text{ ssi } f(w) \in L_B.$$
- On note  $A \leq B$  lorsque  $A$  se réduit à  $B$ .

Attention au sens.

16

## Comparer les problèmes

### Proposition (Réduction)

Si  $A \leq B$ , et si  $B \in P$  (respectivement  $B \in P_{\text{NP}}$ ) alors  $A \in P$  (resp.  $A \in P_{\text{NP}}$ ).

### Proposition (Réduction)

Si  $A \leq B$ , et si  $A \notin P$  (respectivement  $A \notin P_{\text{NP}}$ ) alors  $B \notin P$  (resp.  $B \notin P_{\text{NP}}$ ).

17

## Sous menu

### Problèmes complets

Notion de réduction

#### NP-complétude

Existence de problèmes NP-complet

Premier problème NP-complet naturel

Autres problèmes NP-complets

Sur les réels

18

## NP-complétude

- Un problème  $A$  est dit NP-complet, si
  1. il est dans NP
  2. tout autre problème  $B$  de NP est tel que  $B \leq A$ .
- Autrement dit, un problème NP-complet est un problème maximal pour  $\leq$  parmi les problèmes de la classe NP.

19

## Sous menu

### Problèmes complets

Notion de réduction

NP-complétude

#### Existence de problèmes NP-complet

Premier problème NP-complet naturel

Autres problèmes NP-complets

Sur les réels

20

## Théorème

### Le problème

$K = \{ \langle A, w, \mathbf{1}^n, \mathbf{1}^t \rangle \mid \exists u \in M^*, |u| = n \text{ tel que l'algorithme}$

$A \text{ accepte l'entrée } \langle w, u \rangle \text{ en temps } t \}$

est NP-complet.

21

## Preuve

- Par définition, il s'agit d'un problème de NP.
- Maintenant soit  $L$  un problème de NP.
  - ▶  $L$  est décidé en temps polynomial  $p(n)$  non-déterministe par un algorithme  $A'$ .
  - ▶ Considérer l'algorithme  $A$  qui sur une entrée  $\langle w, u \rangle$  simule  $A'$  en lisant dans les lettres de  $u$  la suite des résultats des interactions avec l'environnement : la  $i$ ème lettre de  $u$  correspond au résultat de la  $i$ ème interaction.
  - ▶ Soit  $p'(n)$  le temps de l'algorithme  $A'$  sur entrée  $\langle w, u \rangle$ , avec  $|w| = n$ .
  - ▶ La fonction qui à  $w$ , de longueur  $n$ , associe  $\langle A', w, \mathbf{1}^{p(n)}, \mathbf{1}^{p'(n)} \rangle$  réalise une réduction de  $L$  vers  $K$ .

22

## Sous menu

### Problèmes complets

Notion de réduction

NP-complétude

Existence de problèmes NP-complet

Premier problème NP-complet naturel

Autres problèmes NP-complets

Sur les réels

23

- On dit qu'un circuit booléen  $C$  est satisfiable, s'il est possible d'affecter ses variables  $\bar{x} = (x_1, \dots, x_n)$  par des éléments de  $\{0, 1\}$ , tel que  $C(\bar{x}) = 1$ .

### Théorème

Étant donné un circuit booléen, le problème de savoir s'il est satisfiable est NP-complet.

24

## Preuve

- Le problème est dans NP, car un circuit  $C$  est satisfiable si et seulement s'il existe un mot  $\bar{x}$  de longueur  $n$ , où  $n$  est le nombre d'entrées du circuit, tel que  $\langle C, \bar{x} \rangle \in \text{CIRCUITVALUE}$ .
- Maintenant soit  $L$  un langage de NP.
  - ▶ Par définition, il existe un problème  $A$  dans P, et un polynôme  $p$ , tel que pour tout mot  $w$ ,  $w \in L$  si et seulement si il existe  $u \in M^*$ ,  $|u| \leq p(|w|)$  avec  $\langle w, u \rangle \in A$ .
  - ▶ Déterminer si  $\langle w, u \rangle \in A$  correspond à un circuit  $C$  de taille polynomiale en  $|\langle w, u \rangle|$ , et donc en  $|w|$ .
  - ▶ Considérons le circuit  $C'$  obtenu en fixant les entrées correspondant au mot  $w$  aux symboles de  $w$ .
- La fonction qui au mot  $w$  associe  $C'$  réalise une réduction de  $L$  vers le problème de satisfiabilité des circuits.

25

## Généralisations : sur $\mathfrak{M} = (M, f_1, \dots, f_u, r_1, \dots, r_v)$

### Théorème

Le problème de la satisfiabilité des circuits à paramètres (on se donne un circuit  $C(\bar{x}, \bar{y})$ , et des paramètres  $\bar{c} \in M^k$  et l'on veut savoir s'il est possible d'affecter les variables  $\bar{y}$  par des éléments de  $M$ , tels que  $C(\bar{c}, \bar{y}) = 1$ ) est  $\text{NP}_{\mathfrak{M}}$ -complet.

### Théorème

Le problème de la satisfiabilité des circuits à paramètres par des entrées booléennes (on se donne un circuit  $C(\bar{x}, \bar{y})$ , et des paramètres  $\bar{c} \in M^k$  et l'on veut savoir s'il est possible d'affecter les variables  $\bar{y}$  par des éléments de  $\{0, 1\}$ , tels que  $C(\bar{c}, \bar{y}) = 1$ ) est  $\text{NDP}_{\mathfrak{M}}$ -complet.

26

## Sous menu

### Problèmes complets

Notion de réduction

NP-complétude

Existence de problèmes NP-complet

Premier problème NP-complet naturel

Autres problèmes NP-complets

Sur les réels

27

### Definition (Formules rudimentaires)

Soit  $\mathfrak{M} = (M, f_1, \dots, f_u, r_1, \dots, r_v)$  une structure. On appelle *formule rudimentaire sur  $\mathfrak{M}$*  une conjonction finie de formules de l'un des types suivants

1.  $x = a$ , où  $a$  est un élément de  $M$ .
2.  $y = f_i(\bar{x})$ , où  $f_i$  est l'une des fonctions de la structure.
3.  $r(\bar{x}) \vee y = 0$ , où  $r_j$  est l'une des relations de la structure.
4.  $\neg r(\bar{x}) \vee y = 1$ , où  $r_j$  est l'une des relations de la structure.
5.  $x = 0 \vee x = 1$ .
6.  $x = \epsilon \vee y = \epsilon'$ , où  $\epsilon, \epsilon' \in \{0, 1\}$ .
7.  $x = \epsilon \vee y = \epsilon' \vee z = \epsilon''$ , où  $\epsilon, \epsilon', \epsilon'' \in \{0, 1\}$ .

### Théorème

Soit  $\mathfrak{M} = (M, f_1, \dots, f_u, r_1, \dots, r_v)$  une structure. Le problème de savoir si une formule rudimentaire est satisfiable est  $\text{NP}_{\mathfrak{M}}$ -complet.

28

## Preuve

- C'est clairement un problème de NP<sub>3SAT</sub>.
- Réciproquement, on montre qu'un circuit est satisfiable si et seulement si une certaine formule rudimentaire de taille polynomiale est satisfaite :
  - ▶ l'astuce est d'introduire une variable par porte du circuit pour représenter la valeur qui en sort (qui peut être un booléen résultat d'un test), et d'écrire que cette variable s'exprime en fonction des entrées de la porte, à l'aide de l'opérateur réalisé par la porte.
  - ▶ On écrit alors la conjonction de toutes les contraintes, et de la contrainte que la sortie doit être 1.
  - ▶ La formule obtenue est de taille polynomiale.

29

## Théorème

*Le problème de la satisfiabilité d'une formule de 3-SAT est NP-complet.*

30

## Preuve

- 3-SAT est dans NP.
- Un algorithme classique correspond à la structure  $\mathfrak{B} = (\{0, 1\}, 0, 1, =)$ .
- $x = y \vee u = 0$  peut être remplacé par  $(x = 0 \vee y = 1 \vee u = 0) \wedge (x = 1 \vee y = 0 \vee u = 0)$ .
- $\neg x = y \vee u = 1$  peut être remplacé par  $(x = 0 \vee y = 0 \vee u = 1) \wedge (x = 1 \vee y = 1 \vee u = 1)$ .
- chaque clause de moins de 3 littéraux peut se remplacer par une conjonction de clauses à 3-littéraux.
  - ▶ Par exemple :  $\alpha \vee \beta$  peut se remplacer par  $(\alpha \vee \beta \vee \gamma) \wedge (\alpha \vee \beta \vee \neg \gamma)$
- $x = 0$  correspond à  $\neg x$ , et  $x = 1$  à  $x$ .

31

## Remarque :

Il n'y a aucune raison que la satisfaction des formules rudimentaires par des entrées booléennes soit NDP<sub>3SAT</sub>-complet puisqu'on introduit des variables comme  $y = f_i(\bar{x})$  qui ne sont pas des booléens.

32

## Sous menu

### Problèmes complets

Notion de réduction

NP-complétude

Existence de problèmes NP-complet

Premier problème NP-complet naturel

Autres problèmes NP-complets

Sur les réels

33

### Théorème

*Dans  $(\mathbb{R}, +, -, *, =, <)$ , le problème de l'existence d'une racine pour un polynôme en  $n$  variables à coefficients réels et de degré total 4 est  $\text{NP}_{(\mathbb{R}, +, -, *, =, <)}$ -complet.*

34

## Au menu

Temps non-déterministe

Problèmes complets

Quelques résultats

35

## Sous menu

### Quelques résultats

Décision vs Construction

Théorèmes de hiérarchie

36

## Théorème

Supposons que  $P = NP$ . Alors pour tout langage  $L \in NP$ , et pour tout vérificateur  $A$  pour  $L$ , il existe un algorithme  $B$  qui sur toute entrée  $w \in L$  produit en temps polynomial un certificat pour  $w$ .

37

## Preuve

- Commençons par le prouver pour  $L$  correspondant au problème de la satisfaction de circuits.
  - ▶ Supposons  $P = NP$  : on peut donc tester si un circuit  $C$  à  $n$ -entrées est satisfiable ou non.
  - ▶ S'il est satisfiable, on peut fixer sa première entrée à 0, et tester si le circuit obtenu  $C_0$  est satisfiable.
  - ▶ Si l'est, on écrit 0 et on recommence récursivement avec ce circuit  $C_0$  à  $n - 1$ -entrées.
  - ▶ Sinon, on écrit 1, et on recommence récursivement avec le circuit  $C_1$  dont la première entrée est fixée à 1, qui possède  $n - 1$ -entrées.
  - ▶ Puisqu'il est facile de vérifier si un circuit sans entrée est satisfiable, par cette méthode, on aura écrit un certificat.
- Maintenant si  $L$  est un langage quelconque de  $NP$ , on peut utiliser le fait que la réduction produite par le théorème précédent est en fait une réduction de Levin :
  - ▶ on peut aussi retrouver un certificat pour  $w$  à partir d'un certificat de la satisfiabilité du circuit  $f(w)$ .
  - ▶ On peut donc utiliser l'algorithme précédent pour retrouver un certificat pour  $L$ .

38

## Sous menu

### Quelques résultats

Décision vs Construction

Théorèmes de hiérarchie

39

On dit qu'une fonction  $f(n)$  est *constructible en temps*, si la fonction  $w \mapsto f(|w|)$  est calculable en temps  $O(f(n))$ .

### Théorème (Théorème de hiérarchie)

Soient  $f, f' : \mathbb{N} \rightarrow \mathbb{N}$  des fonctions constructibles en temps tel que  $f(n) \log(f(n)) = o(f'(n))$ . Alors l'inclusion  $\text{TIME}(f) \subset \text{TIME}(f')$  est stricte.

### Théorème (Théorème de hiérarchie)

Soient  $f, f' : \mathbb{N} \rightarrow \mathbb{N}$  des fonctions constructibles en temps tel que  $f(n + 1) = o(f'(n))$ . Alors l'inclusion  $\text{NTIME}(f) \subset \text{NTIME}(f')$  est stricte.

Se généralise au temps non-déterministe.

40

## Conséquences

- Par exemple :

Corollaire

$$\text{TIME}(n^2) \subsetneq \text{TIME}(n^{\log n}) \subsetneq \text{TIME}(2^n).$$

- On définit :

$$\text{EXPTIME} = \bigcup_{c \in \mathbb{N}} \text{TIME}(2^{n^c})$$

Corollaire

$$P \subsetneq \text{EXPTIME}.$$

41

## Preuve

- Tout polynôme devient ultimement négligeable devant  $2^n$ , et donc  $P$  est un sous-ensemble de  $\text{TIME}(2^n)$ .
- Maintenant  $\text{TIME}(2^n)$ , qui contient tout  $P$  est un sous-ensemble strict de, par exemple,  $\text{TIME}(2^{n^2})$ .

42