Modèle de Blum, Shub et Smale
Théorème de transfert pour P = NP dans $\mathbb{R}$

Fabrice Ben Hamouda

Cours de Langages formels, Calculabilité et Complexité

Janvier 2010

1. Présentation du modèle
   - Machines de Blum, Shub et Smale (BSS)
   - Lien avec les machines standards
   - Classes de complexité

2. $\mathrm{NP}_{\mathbb{R}} = \mathrm{NBP}_{\mathbb{R}}$ dans $(\mathbb{R}, +, -, =, <)$
   - Remarques préliminaires
   - Un lemme d'algèbre linéaire
   - $\mathrm{NP}_{\mathbb{R}} = \mathrm{NBP}_{\mathbb{R}}$ dans $(\mathbb{R}, +, -, =, <)$

3. Un théorème de transfert
   - $\mathrm{P}^0_{\mathbb{R}} = \mathrm{NP}^0_{\mathbb{R}} \Longrightarrow \mathrm{P} = \mathrm{NP}$
   - $\mathrm{P} = \mathrm{NP} \Longrightarrow \mathrm{P}^0_{\mathbb{R}} = \mathrm{NP}^0_{\mathbb{R}}$
   - Localisation d'un point

Introduction

Les machines de Blum, Shub et Smale (BSS) sont des machines de Turing capable de calculer dans une structure $\mathbf{M}$ quelconque.

Définition

Une structure est un ensemble muni de fonctions et de relations.

Introduction

Exemple

- $(\{0,1\},=)$:
  - *fonctions* : identité
  - *relations* : égalité $(=)$

Introduction

Exemple

- $(\{0,1\},=)$:
  - *fonctions* : identité
  - *relations* : égalité $(=)$
- $(\mathbb{R},+,-,=,<)$
  - *fonctions* : identité, addition $(+)$, passage à l'opposé $(-)$
  - *relations* : égalité $(=)$, ordre strict $(<)$

Introduction

Exemple

- $(\{0,1\},=)$:
  - *fonctions* : identité
  - *relations* : égalité $(=)$
- $(\mathbb{R},+,-,=,<)$
  - *fonctions* : identité, addition $(+)$, passage à l'opposé $(-)$
  - *relations* : égalité $(=)$, ordre strict $(<)$

Attention

Les éléments de la structure sont des entités atomiques.
Par exemple, dans $(\mathbb{R},+,-,=,<)$, il n'y a pas de problème d'arrondis.

Introduction

Question P = NP

Introduction

Question P = NP

- Dans $(\mathbb{R}, +, -, =)$:

$$\mathrm{P}_{\mathbb{R}} \neq \mathrm{NP}_{\mathbb{R}}$$

Question P = NP

- Dans $(\mathbb{R}, +, -, =)$:

$$\mathrm{P}_{\mathbb{R}} \neq \mathrm{NP}_{\mathbb{R}}$$

- Il existe une structure (non triviale) dans laquelle :

$$\mathrm{P} = \mathrm{NP}$$

Question P = NP

- Dans $(\mathbb{R}, +, -, =)$:

$$\mathrm{P}_{\mathbb{R}} \neq \mathrm{NP}_{\mathbb{R}}$$

- Il existe une structure (non triviale) dans laquelle :

$$\mathrm{P} = \mathrm{NP}$$

- La question P = NP est ouverte dans les structures suivantes : $(\mathbb{R}, +, -, =, <)$, $(\mathbb{R}, +, \times, =, <)$ et $(\mathbb{C}, +, \times, =)$.

Introduction

L'objectif de cet exposé est de montrer le théorème de transfert suivant :

Théorème

Dans la structure $(\mathbb{R}, +, -, =, <)$*, on a :*

$$\mathrm{P}^0_{\mathbb{R}} = \mathrm{NP}^0_{\mathbb{R}}$$

si et seulement si

$$\mathrm{P} = \mathrm{NP}^1$$

[1] au sens standard

Plan

Machines de Blum, Shub et Smale (BSS) I

Fixons une structure $\mathbf{M}$, qui contient la fonction identité et les constantes 0 et 1.

Une machine BSS est une machine de Turing :

- à plusieurs bandes bi-infinies
- d'alphabet d'entrée : $\mathbf{M}$
- d'alphabet de bande : $\mathbf{M} \cup \{\#\}$
- possédant un nombre fini p de constantes : $c_1, \ldots, c_p$ appelés **paramètres**
- commandée par une liste finie d'instructions

Machines de Blum, Shub et Smale (BSS) II

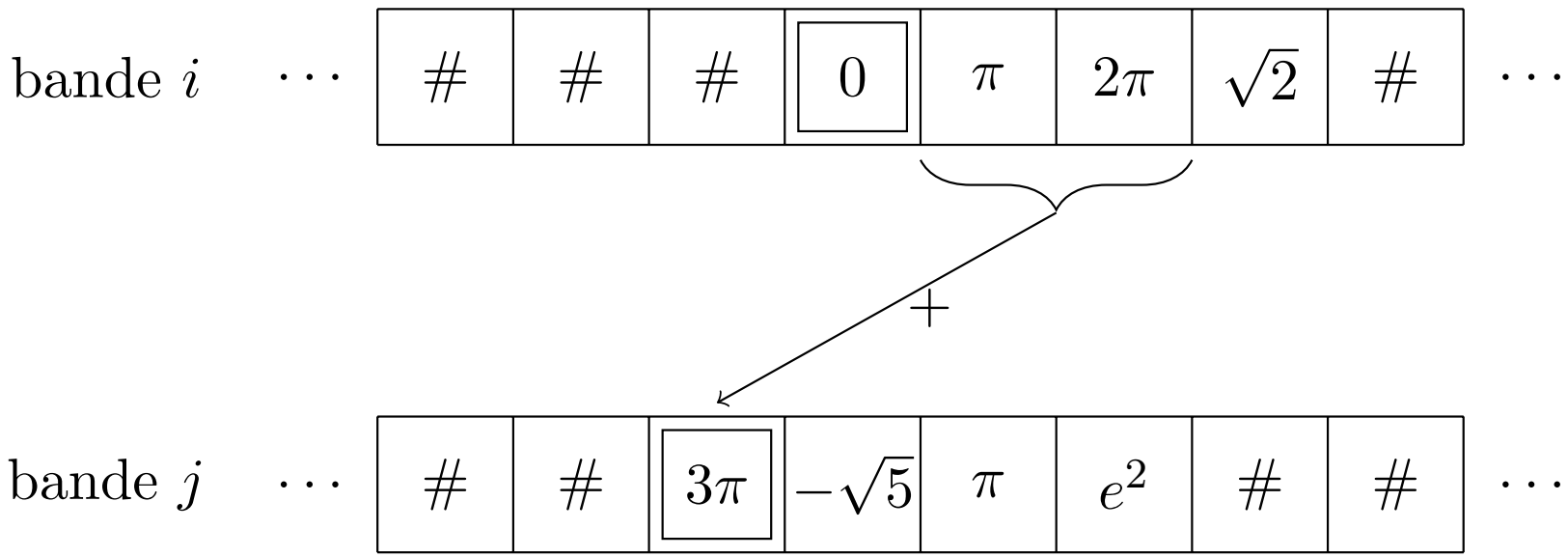
On suppose de plus que :

- une machine BSS s'arrête sur chaque entrée.
- 1 est une des constantes.
- le résultat du calcul est écrit sur la dernière bande (si le problème considéré est un problème de décision, la sortie est soit 1 pour *vrai*, soit 0 pour *faux*).

Instructions d'une machine BSS I

- **stop** : arrêter la machine.
- **déplace**(i,d) : déplacer la tête de lecture de la $i^{ème}$ bande à gauche (si $d = -1$) ou à droite (si $d = 1$).
- **ecrit**(i,j) : écrit le paramètre c_j ou 0 (si $j = -1$) ou # (si $j = -2$) dans la case pointée par la tête de lecture de la $i^{ème}$ bande.

Instructions d'une machine BSS II



- **calcule(f,i,j)** : f étant une fonction n-aire de la structure, lire les n valeurs $a_1, ..., a_n$ immédiatement à droite de la tête de lecture de la $i^{ème}$ bande, calculer $f(a_1, ..., a_n)$ et stocker le résultat dans la case pointée par la tête de lecture de la $j^{ème}$ bande. Comme l'identité est une fonction du langage, il est ainsi possible de copier une case dans une autre.

Instructions d'une machine BSS III

- **test**(r,i,q,q') : r étant une relation n-aire de la structure, lire les n valeurs $a_1, ..., a_n$ immédiatement à droite de la tête de lecture de la $i^{ème}$ bande et si ces valeurs satisfont la relation r, aller à l'instruction q sinon aller à l'instruction q'.
- **testvide**(i,q,q') : si la case pointée par la tête de lecture de la $i^{ème}$ bande est vide, aller à l'instruction q sinon aller à l'instruction q'.

Lien avec les machines standards I

Proposition

Une machine de Turing classique qui s'exécute en temps polynomial peut être transformée en une machine de Turing classique avec comme alphabet d'entrée $\{0, 1\}$ *et comme alphabet de bande* $\{0, 1, \#\}$ *qui s'exécute en temps polynomial.*

Lien avec les machines standards II

Proposition

Pour la structure $(\{0,1\},=)$*, les machines* BSS *sont équivalentes aux machines de Turing* ***déterministes*** *classiques (et qui s'arrêtent sur toutes les entrées) avec comme alphabet d'entrée* $\{0,1\}$ *et comme alphabet de bande* $\{0,1,\#\}$*.*
De plus le temps d'un calcul de taille n *sur la machine transformée (dans un sens comme dans l'autre) dépend polynomialement du temps de calcul sur la machine initiale.*

Classes de complexité I

Notons P et NP les classes de complexité standard.

Notons $\mathbb{R}^\infty$ l'ensemble des mots sur l'alphabet $\mathbb{R}$ et $\{0,1\}^*$ l'ensemble des mots binaires.

Définissons les classes de complexité pour les machines BSS :

Définition

Un problème (c'est-à-dire un langage) L est dans $\mathrm{P}_\mathbb{R}$ si et seulement s'il existe une machine BSS *qui détermine si un mot x de taille n est dans le langage L en un temps majoré par un polynôme en n.*

Classes de complexité II

Définition

Un problème L est dans $\mathrm{NP}_{\mathbb{R}}$ si et seulement si :

- *il existe un langage L' tel que $L = \{x \in \mathbb{R}^\infty \mid \exists y \in \mathbb{R}^\infty, (x, y) \in L'\}$ et tel que la taille maximale d'un y pour un x de taille n est majorée par un polynôme de taille n.*
- *et il existe une machine* BSS *qui prend en entrée un couple $< x, y >$ et qui détermine si le mot (x, y) est dans le langage L' en un temps polynomial en n, la taille de x.*

Le y (qui n'est pas forcément unique) associé à un x est appelé certificat d'appartenance de x à L.

Classes de complexité III

Définition

Un problème L est dans $\mathrm{NBP}_{\mathbb{R}}$ *si et seulement s'il est dans* $\mathrm{NP}_{\mathbb{R}}$ *et qu'on peut n'utiliser que des certificats booléens (ou binaires - c'est-à-dire dans* $\{0,1\}^*$*).*

On définit $\mathrm{P}^0_{\mathbb{R}}$, $\mathrm{NP}^0_{\mathbb{R}}$ et $\mathrm{NBP}^0_{\mathbb{R}}$ comme $\mathrm{P}_{\mathbb{R}}$, $\mathrm{NP}_{\mathbb{R}}$ et $\mathrm{NBP}_{\mathbb{R}}$ sauf que l'on impose aussi que la machine BSS considérée n'ait que 1 comme paramètre.

Remarques sur les classes $\mathrm{NP}_{\mathbb{R}}(\mathrm{NP}^0_{\mathbb{R}})$ et $\mathrm{NBP}_{\mathbb{R}}(\mathrm{NBP}^0_{\mathbb{R}})$

Remarque

Dans toute structure, on a :

$$\mathrm{NBP}_{\mathbb{R}} \subset \mathrm{NP}_{\mathbb{R}}$$

et :

$$\mathrm{NBP}^0_{\mathbb{R}} \subset \mathrm{NP}^0_{\mathbb{R}}$$

Plan

Objectif

L'objectif de cette section est de montrer les deux théorèmes suivants :

Théorème

Dans la structure $(\mathbb{R}, +, -, =, <)$, *on a :*

$$\mathrm{NP}_{\mathbb{R}} = \mathrm{NBP}_{\mathbb{R}}$$

Théorème

Dans la structure $(\mathbb{R}, +, -, =, <)$, *on a :*

$$\mathrm{NP}^0_{\mathbb{R}} = \mathrm{NBP}^0_{\mathbb{R}}$$

Idée de la démonstration

Considérons un langage L de $\mathrm{NP}_{\mathbb{R}}$.
Remplaçons pour chaque mot du langage $(x_1, \ldots, x_n)$, son certificat réel $(y_1, \ldots, y_m)$ par une matrice $(\tilde{y}_{i,j})$ de rationnels (codés en binaire) qui représente :

$$(\tilde{y}_1, \ldots, \tilde{y}_m) = \left(\sum_{j=1}^{n} \tilde{y}_{1,j} x_j + \sum_{j=1}^{p} \tilde{y}_{1,n+j} c_j, \ldots, \sum_{j=1}^{n} \tilde{y}_{m,j} x_j + \sum_{j=1}^{p} \tilde{y}_{m,n+j} c_j \right)$$

et de telle sorte que :

$$y_i \approx \tilde{y}_i = \sum_{j=1}^{n} \tilde{y}_{i,j} x_i + \sum_{j=1}^{p} \tilde{y}_{i,n+j} c_j$$

au sens où remplacer y_i par $\sum_{j=1}^{n} \tilde{y}_{i,j} x_i + \sum_{j=1}^{p} \tilde{y}_{i,n+j} c_j$ ne va pas changer le comportement de la machine BSS chargée de vérifier le certificat.

Remarques préliminaires I

Définition

On appelle taille d'un entier n la partie entière supérieure de $\log_2 |n|$. C'est le nombre de chiffres de l'entier écrit en binaire (sans le signe). La taille d'un rationnel est le maximum de la taille de son numérateur et de son dénominateur quand il est écrit sous forme irréductible.

Un entier peut être représenté par la suite des bits de sa valeur absolue (suite de taille au plus n) plus un bit pour le signe.
Un rationnel peut être représenté comme un couple de deux entiers (numérateur et dénominateur).

Remarques préliminaires II

À chaque étape, une machine BSS :

- ajoute deux cases
- ou passe à l'opposé une case
- ou écrit un paramètre (ou 0 ou #) dans une case
- ou fait autre chose, ce qui ne modifie pas ses bandes

On peut remarquer qu'à une étape t' du calcul d'une machine BSS sur une entrée $(x_1, \ldots, x_n)$, les cases mémoires sont des combinaisons linéaires des paramètres $(c_1, \ldots, c_p)$ et des entrées $(x_1, \ldots, x_n)$ à coefficients entiers de taille au plus $t' + 1$. Cela se montre aisément par récurrence, car la somme de deux entiers de taille L est de taille au plus $L + 1$.

Remarques préliminaires III

D'où la proposition suivante :

Proposition

Les comparaisons effectuées par une machine BSS *sur une entrée* $(x_1, \ldots, x_n)$ *en un temps* t *sont de la forme (quitte à changer les termes de membre) :*

- *ou bien* $\lambda_1 x_1 + \ldots + \lambda_n x_n = \mu_1 c_1 + \ldots + \mu_p c_p$
- *ou bien* $\lambda_1 x_1 + \ldots + \lambda_n x_n < \mu_1 c_1 + \ldots + \mu_p c_p$

avec les λ_i *et les* μ_j *des entiers de taille au plus* t*.*

Un lemme d'algèbre linéaire

Lemme

Considérons un système S de la forme (d'inconnues $x_1, \ldots, x_n$) :

$$\begin{cases} \lambda_{1,1}x_1 + \ldots + \lambda_{1,n}x_n & \geq & a_1 \\ \vdots & \vdots & \vdots \\ \lambda_{m,1}x_1 + \ldots + \lambda_{m,n}x_n & \geq & a_m \\ \mu_{1,1}x_1 + \ldots + \mu_{1,n}x_n & > & b_1 \\ \vdots & \vdots & \vdots \\ \mu_{p,1}x_1 + \ldots + \mu_{p,n}x_n & > & b_p \end{cases}$$

*où $a_i, b_j \in \mathbb{R}$ pour tout i, j et où $\lambda_{i,j}, \mu_{i,j}$ sont des **entiers**. Ce système a une solution sous la forme d'un vecteur de combinaisons linéaires d'au plus $2n+1$ des a_k et b_k, et à coefficients rationnels pas trop grands :*

$$x_i = \alpha_1 a_1 + \ldots + \alpha_m a_m + \beta_1 b_1 + \ldots + \beta_p b_p$$

$\mathrm{NP}_{\mathbb{R}} = \mathrm{NBP}_{\mathbb{R}}$ dans $(\mathbb{R}, +, -, =, <)$

Montrons que : $\mathrm{NP}_{\mathbb{R}} \subset \mathrm{NBP}_{\mathbb{R}}$.
Soit L un langage de $\mathrm{NP}_{\mathbb{R}}$, $\vec{x} = (x_1, \ldots, x_n)$ un mot du langage L et $\vec{y} = (y_1, \ldots, y_m)$ un de ses certificats.
Les tests effectués par la machine sont de la forme :

- ou bien $\lambda_{i,1} y_1 + \ldots + \lambda_{i,m} y_m = a_i$
- ou bien $\lambda_{i,1} y_1 + \ldots + \lambda_{i,m} y_m < a_i$

avec les $\lambda_{i,j}$ des **entiers** de taille au plus t et les a_i des combinaisons linéaires des x_k et des paramètres c_k de la machine.

$\mathrm{NP}_{\mathbb{R}} = \mathrm{NBP}_{\mathbb{R}}$ dans $(\mathbb{R}, +, -, =, <)$

On construit un système S en ajoutant, pour chaque $(m+1)$-uplet $(\lambda_{i,1}, \ldots, \lambda_{i,m}, a_i)$ qui apparaît dans un test de la machine, l'équation (ou inéquation) vérifiée par $\vec{y}$ parmi les trois suivantes :

- $\lambda_{i,1} y_1 + \ldots + \lambda_{i,m} y_m < a_i$
- $\lambda_{i,1} y_1 + \ldots + \lambda_{i,m} y_m = a_i$
- $\lambda_{i,1} y_1 + \ldots + \lambda_{i,m} y_m > a_i$

$\mathrm{NP}_{\mathbb{R}} = \mathrm{NBP}_{\mathbb{R}}$ dans $(\mathbb{R}, +, -, =, <)$

On construit un système S en ajoutant, pour chaque $(m+1)$-uplet $(\lambda_{i,1}, \ldots, \lambda_{i,m}, a_i)$ qui apparaît dans un test de la machine, l'équation (ou inéquation) vérifiée par $\vec{y}$ parmi les trois suivantes :

- $\lambda_{i,1} y_1 + \ldots + \lambda_{i,m} y_m < a_i$
- $\lambda_{i,1} y_1 + \ldots + \lambda_{i,m} y_m = a_i$
- $\lambda_{i,1} y_1 + \ldots + \lambda_{i,m} y_m > a_i$

On peut trouver une solution de ce système sous la forme :

$$\vec{\tilde{y}} = \left(\sum_{j=1}^{n} \tilde{y}_{1,j} x_i + \sum_{j=1}^{p} \tilde{y}_{1,n+j} c_j, \ldots, \sum_{j=1}^{n} \tilde{y}_{m,j} x_j + \sum_{j=1}^{p} \tilde{y}_{m,n+j} c_j \right)$$

On donne alors comme certificat booléen la matrice $(\tilde{y}_{i,j})$. □

$\mathrm{NP}^0_{\mathbb{R}} = \mathrm{NBP}^0_{\mathbb{R}}$ dans $(\mathbb{R}, +, -, =, <)$

La démonstration de $\mathrm{NP}^0_{\mathbb{R}} = \mathrm{NBP}^0_{\mathbb{R}}$ dans $(\mathbb{R}, +, -, =, <)$ est quasiment identique.

Montrons maintenant le théorème de transfert annoncé.

Plan

Énoncé

Théorème

Dans la structure $(\mathbb{R}, +, -, =, <)$, *on a :*

$$\mathrm{P}^0_{\mathbb{R}} = \mathrm{NP}^0_{\mathbb{R}}$$

si et seulement si

$$\mathrm{P} = \mathrm{NP}$$

Démonstration.
Supposons $\mathrm{P}^0_{\mathbb{R}} = \mathrm{NP}^0_{\mathbb{R}}$ et montrons que $\mathrm{NP} \subset \mathrm{P}$.

Considérons un langage booléen L dans NP. Soit $\mathcal{M}$ le vérificateur BSS de certificats.
L est aussi dans $\mathrm{NP}^0_{\mathbb{R}}$ car il est aisé de transformer une machine de Turing classique en une machine BSS (n'utilisant que le paramètre 1) en multipliant le temps d'exécution par une constante. Donc L est un langage $\mathrm{P}^0_{\mathbb{R}}$ et il existe une machine BSS $\mathcal{M}$ qui s'exécute en temps polynomial et qui reconnaît L.
Il existe donc une machine de Turing classique qui reconnaît L en temps polynomial et $L \in \mathrm{NP}$. □

Idée de la démonstration de la réciproque

Supposons P = NP.

Considérons un problème L de $\mathrm{NBP}^0_{\mathbb{R}}$ (car $\mathrm{NBP}^0_{\mathbb{R}} = \mathrm{NP}^0_{\mathbb{R}}$).
On montre que l'on peut calculer, pour tout vecteur de réels $\vec{x} = (x_1, \ldots, x_n)$, un vecteur de rationnels

$$\vec{\tilde{x}} = (\tilde{x}_1, \ldots, \tilde{x}_n)$$

suffisamment proche de $\vec{x}$ pour que :

$$\vec{x} \in L \text{ si et seulement si } \vec{\tilde{x}} \in L$$

Comme les certificats utilisés sont binaires, il existe une machine capable de dire en temps polynomial (car on a supposé P = NP) si $\vec{\tilde{x}} \in L$ ou non.

Définition

Un oracle (booléen de NP*) est un mécanisme qui permet à une machine de Turing classique ou* BSS *d'interroger à un coût unitaire un langage L fixé de classe* NP*, autrement dit de savoir si un mot écrit sur une de ses bandes est dans le langage L ou non.*

Remarque

Comme dans toute cette sous-section et la suivante, on suppose $\mathrm{NP} = \mathrm{P}$*, on peut utiliser des oracles* NP *dans toute machine de Turing classique ou* BSS *(mais l'entrée de l'oracle est toujours composée de* 0 *et de* 1*).*

$\mathrm{P} = \mathrm{NP} \Longrightarrow \mathrm{P}^0_{\mathbb{R}} = \mathrm{NP}^0_{\mathbb{R}}$

Considérons un problème L de $\mathrm{NBP}^0_{\mathbb{R}}$ (car $\mathrm{NBP}^0_{\mathbb{R}} = \mathrm{NP}^0_{\mathbb{R}}$).
Plaçons nous dans l'espace affine $\mathbb{R}^n$.
Soit $\vec{x} = x = (x_1, \ldots, x_n)$ un mot de L.

Soit $\mathcal{H}_n = \{h_1, \ldots, h_m\}$ l'ensemble des hyperplans affines d'équations $\lambda_1 x_1 + \ldots + \lambda_n x_n = b$ avec λ_i et b des entiers de taille au plus t.
Chaque hyperplan h_i d'équation $\lambda_1 x_1 + \ldots + \lambda_n x_n = b$ définit deux *demi-espaces* :

- h_i^+ d'équation : $\lambda_1 x_1 + \ldots + \lambda_n x_n > b$
- h_i^- d'équation : $\lambda_1 x_1 + \ldots + \lambda_n x_n < b$

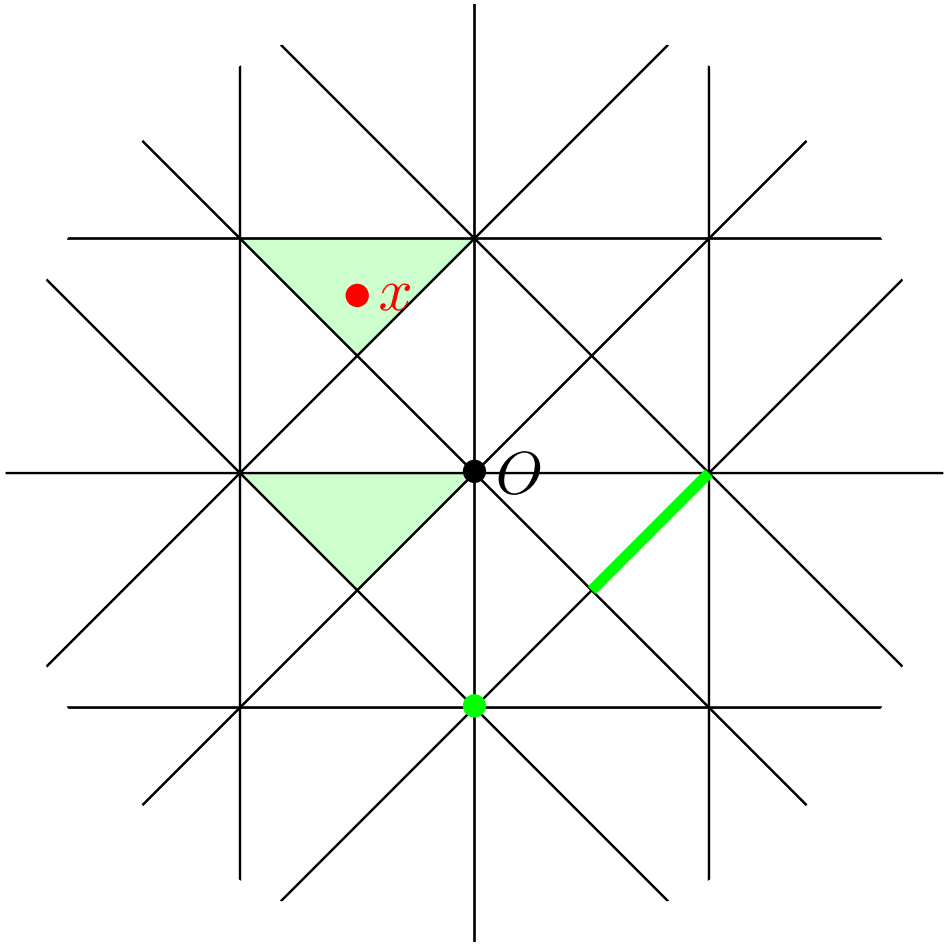
Cellules

On définit une relation d'équivalence $\sim$ sur les points de $\mathbb{R}^n$ de la façon suivante :

$$x' \sim x'' \iff \forall i \in [\![1, m]\!], \begin{cases} x'' \in h_i^+ & \text{si} \quad x' \in h_i^+ \\ x'' \in h_i & \text{si} \quad x' \in h_i \\ x'' \in h_i^- & \text{si} \quad x' \in h_i^- \end{cases}$$

Les classes d'équivalence de cette relation sont appelées des *cellules*.

Cellules en dimension 2



Localisation du point x

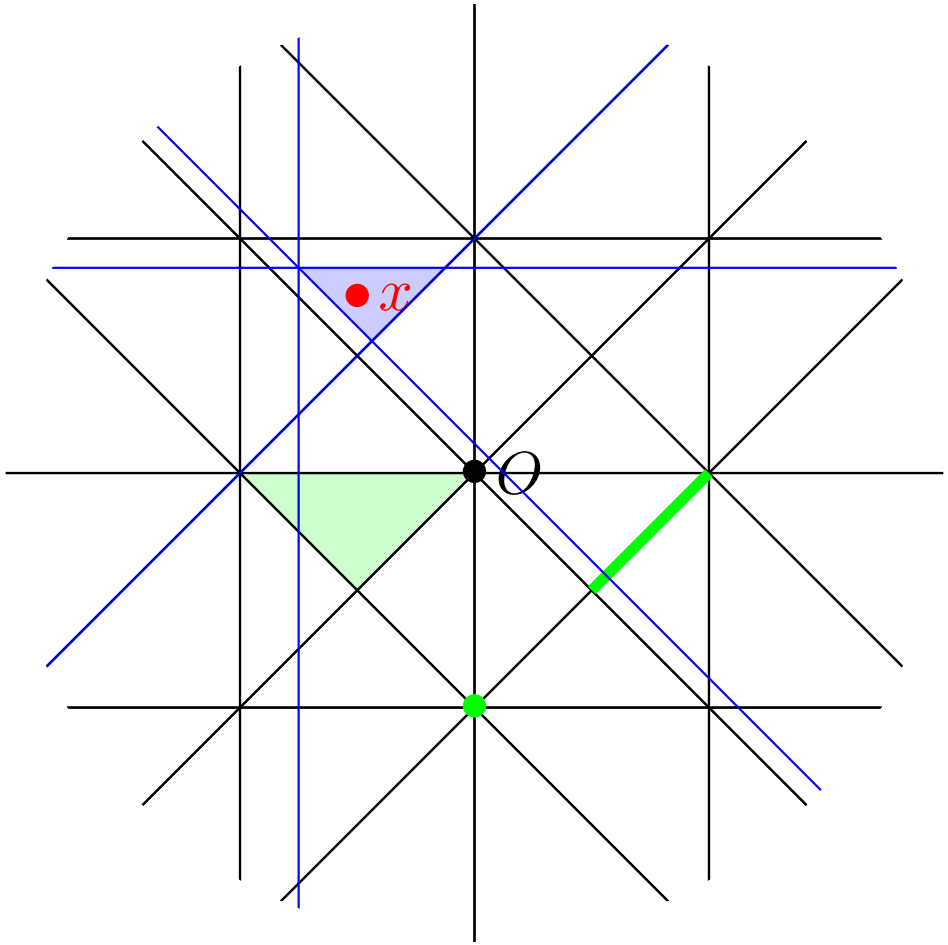
Chaque classe d'équivalence est soit toute entière incluse dans le langage L soit d'intersection vide avec L. En effet les éléments d'une même classe d'équivalence ont les mêmes résultats à chacun des tests de la machine $\mathcal{M}$.

Problème de localisation

Trouver un système d'équations dont la solution est incluse dans la cellule de x.

Supposons que nous possédons une machine BSS capable de résoudre ce problème en temps polynomial.

Cellules en dimension 2



Preuve de $\mathrm{P} = \mathrm{NP} \Longrightarrow \mathrm{P}^0_{\mathbb{R}} = \mathrm{NP}^0_{\mathbb{R}}$

Démonstration.

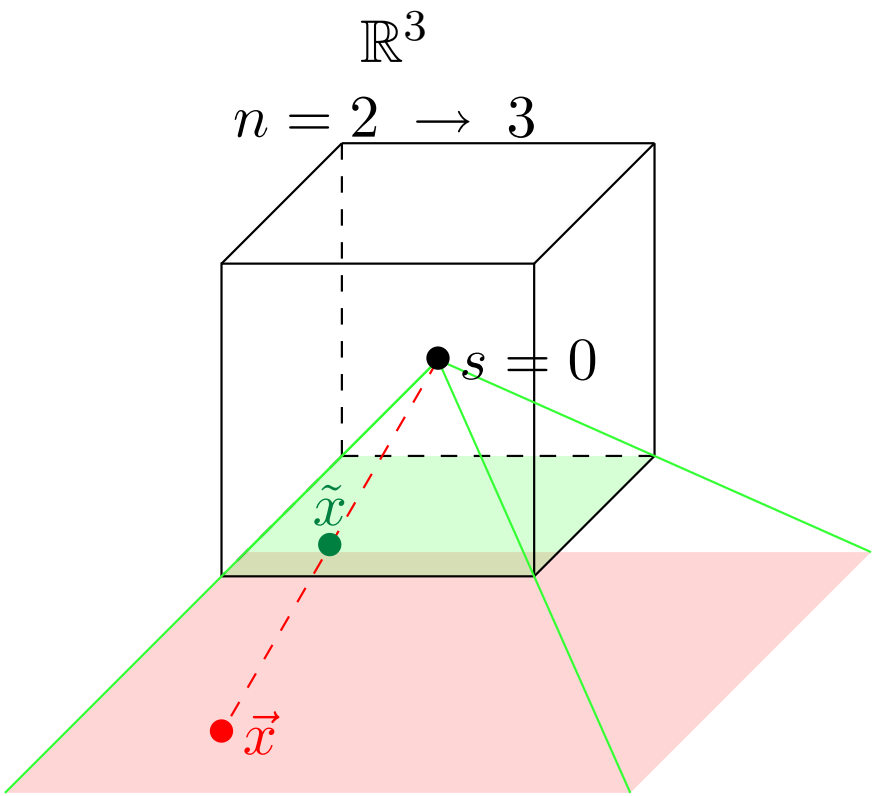
- D'après le lemme d'algèbre linéaire précédent, il existe une solution rationnelle au système précédent à coefficients de taille polynomiale en n.
- De plus : $x \in L$ si et seulement si $\tilde{x} \in L$.

L'algorithme suivant convient donc :

1. Localiser le point x.
2. Trouver (en la devinant par un oracle NP) une solution rationnelle $\tilde{x}$ du système obtenu.
3. Regarder si $\tilde{x} \in L$ convient, grâce à un oracle NP.

□

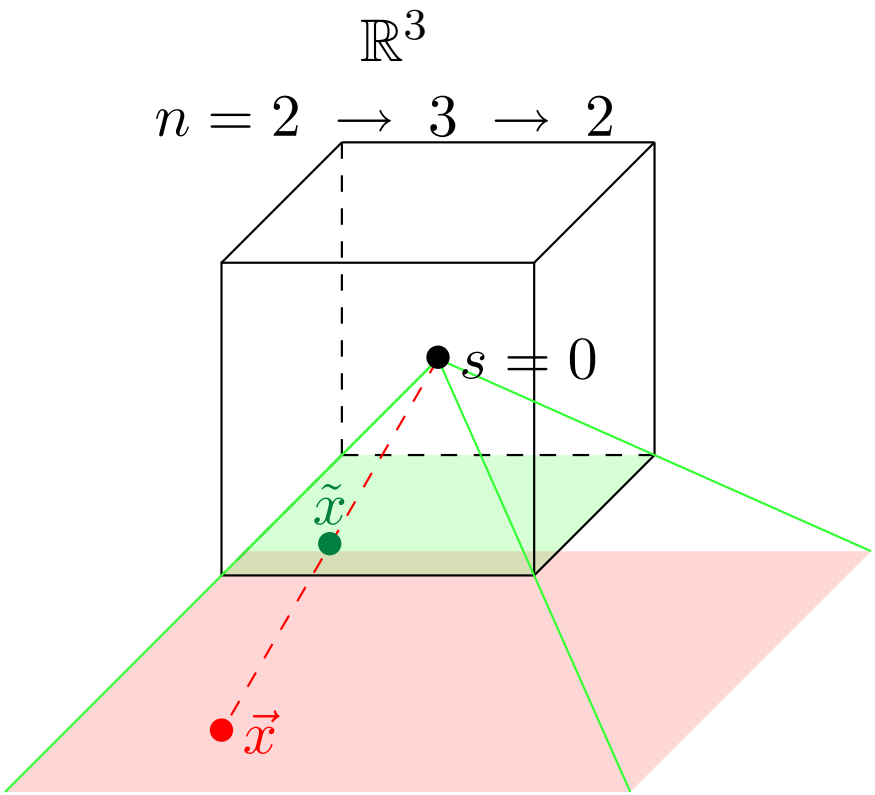
Localisation d'un point



$$x = (x_1, x_2) \in \mathbb{R}^2$$
$$\downarrow$$
$$\vec{x} = (x_1, x_2, 0) \in \mathbb{R}^3$$

1. Plonger l'espace affine dans l'espace vectoriel $\mathbb{R}^{n+1}$.

Localisation d'un point



$$\vec{x} = (x_1, x_2, 0) \in \mathbb{R}^3$$

$$\downarrow$$

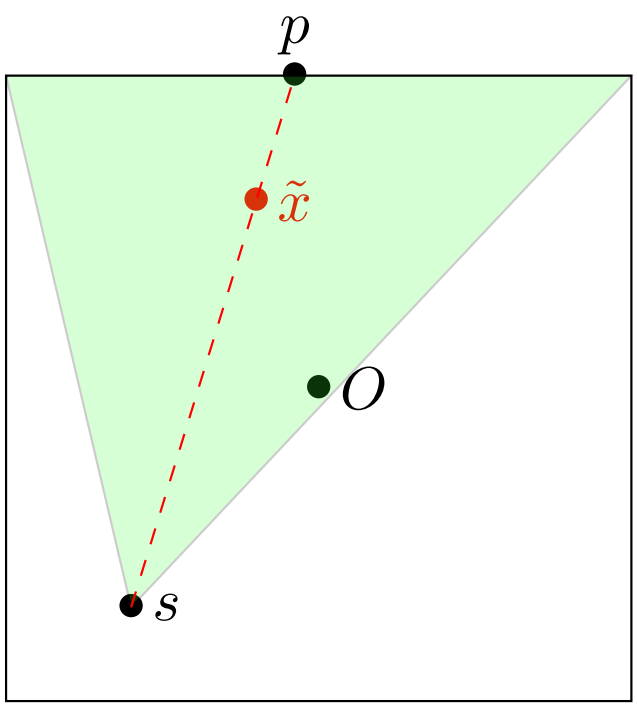
$\tilde{x} = (\tilde{x}_1, \tilde{x}_2, \tilde{x}_3)$ sur une face f du cube $[-1, 1]^3$

1. Plonger l'espace affine dans l'espace vectoriel $\mathbb{R}^{n+1}$.
2. Projeter x selon la droite (Ox) sur la bonne face de l'hypercube $[-1, 1]^{n+1}$.

Localisation d'un point

Face f d'un cube $(\mathbb{R}^2)$

$n = 2 \rightarrow 3 \rightarrow 2 \rightarrow 1$



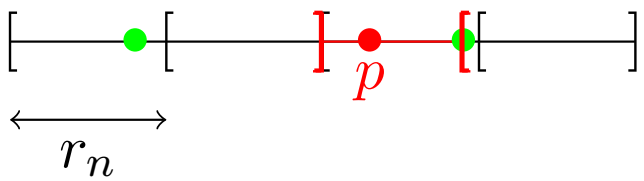
$\tilde{x} \in f$

$\downarrow$

p sur une arête du carré $[-1, 1]^2$

1. Plonger l'espace affine dans l'espace vectoriel $\mathbb{R}^{n+1}$.
2. Projeter x selon la droite (Ox) sur la bonne face de l'hypercube $[-1, 1]^{n+1}$.
3. Diminuer par projections successives la dimension de l'espace.

Localisation d'un point



1. Plonger l'espace affine dans l'espace vectoriel $\mathbb{R}^{n+1}$.
2. Projeter x selon la droite (Ox) sur la bonne face de l'hypercube $[-1,1]^{n+1}$.
3. Diminuer par projections successives la dimension de l'espace.
4. En dimension 1, une recherche dichotomique suffit.

Références I

B. Poizat.
Les petits cailloux.
Aléas, 1995.

H. Fournier and P. Koiran.
Lower bounds are not easier over the reals : Inside PH.
Lecture notes in computer science, pages 832–843, 1999.

L. Blum, M. Shub, and S. Smale.
On a theory of computation and complexity over the real numbers : NP-completeness, recursive functions and universal machines.
American Mathematical Society, 21(1), 1989.

P. Koiran.
Computing over the reals with addition and order.
Theoretical Computer Science, 133(1) :35–47, 1994.

Références II


Friedhelm Meyer auf der Heide.
Fast algorithms for n-dimensional restrictions of hard problems.
J. ACM, 35(3) :740–747, 1988.


C. Gaßner.
Eine Struktur endlicher Signatur mit Identitätsrelation und mit P= NP.
2004.


A. Hemmerling.
P= NP for some structures over the binary words.
Journal of Complexity, 21(4) :557–578, 2005.

Je remercie chaleureusement Monsieur Sylvain Perifel pour son aide précieuse à l'élaboration de ce dossier et pour sa relecture.