Complexité

Paul Gastin
Paul.Gastin@liafa.jussieu.fr

http://liafa.jussieu.fr/∼gastin/MMFAI/MMFAI.html

LIAFA, Université Paris 7, UMR CNRS 7089

4 Complexité : plan

1. Introduction
2. Classes de complexité
3. Réduction et complétude
4. Problème SAT
5. Circuits booléens
6. Relations entre classes de complexité

4.1 Introduction

Combien de temps / d'espace faut-il pour résoudre un problème.

Le modèle de calcul est la machine de Turing (MT).

La complexité dépend uniquement du problème et pas d'un algorithme particulier pour résoudre le problème.

Comme pour la calculabilité, un problème (binaire) est défini par
– L'ensemble (dénombrable) de ses données,
– la description de la représentation de ces données.
– Un énoncé portant sur ces données pouvant être vrai ou faux.

4.2 Classes de complexité

TIME(f) classe des problèmes que l'on peut résoudre en temps f par une MT déterministe.

NTIME(f) classe des problèmes que l'on peut résoudre en temps f par une MT non déterministe.

co-NTIME(f) classe des problèmes dont le complémentaire peut se résoudre en temps f par une MT non déterministe.

SPACE(f) classe des problèmes que l'on peut résoudre en espace f par une MT déterministe.

NSPACE(f) classe des problèmes que l'on peut résoudre en espace f par une MT non déterministe.

4.2 Classes importantes en temps

$$\mathbf{P} = \bigcup_{k \geq 0} \mathbf{TIME}(n^k)$$

$$\mathbf{NP} = \bigcup_{k \geq 0} \mathbf{NTIME}(n^k)$$

$$\mathbf{co\text{-}NP} = \bigcup_{k \geq 0} \mathbf{co\text{-}NTIME}(n^k)$$

$$\mathbf{EXP} = \bigcup_{k \geq 0} \mathbf{TIME}(2^{n^k})$$

$$\mathbf{NEXP} = \bigcup_{k \geq 0} \mathbf{NTIME}(2^{n^k})$$

4.2 Classes importantes en espace

$$\mathbf{L} = \mathbf{SPACE}(\log n)$$

$$\mathbf{NL} = \mathbf{NSPACE}(\log n)$$

$$\mathbf{PSPACE} = \bigcup_{k \geq 0} \mathbf{SPACE}(n^k)$$

4.3 Réduction et complétude

Définition 4.1
Un problème $K_1 \in \Sigma_1^*$ se réduit à un problème $K_2 \subseteq \Sigma_2^*$
en espace logarithmique (resp. en temps polynomial)
s'il existe une fonction $f : \Sigma_1^* \to \Sigma_2^*$ calculable par une MT déterministe
en espace logarithmique (resp. en temps polynomial)
telle que $\forall x \in \Sigma_1^*$, $x \in K_1 \iff f(x) \in K_2$.

Proposition 4.2 *Soit $f : \Sigma_1^* \to \Sigma_2^*$ une fonction calculée en espace logarithmique par une MT déterministe M. La MT M travaille en temps polynomial et il existe p tel que $\forall x \in \Sigma_1^*$, $|f(x)| = \mathcal{O}(|x|^p)$.*

Proposition 4.3
La composée de deux réductions logarithmiques (resp. polynomiales) est encore une réduction logarithmique (resp. polynomiale).

4.3 Réduction et complétude

Définition 4.4 *Une classe de complexité $\mathcal{C}$ est fermée par réduction si K_1 se réduit à K_2 et $K_2 \in \mathcal{C}$ impliquent $K_1 \in \mathcal{C}$.*

Proposition 4.5
Les classes **L** *et* **NL** *sont fermées par réduction logarithmique.*
Les classes **P**, **NP**, **EXP**, **NEXP**, **PSPACE**, *. . . sont fermées par réduction polymiale.*

Applications : Pour montrer qu'un problème K_1 est dans une classe $\mathcal{C}$ fermée par réduction, il suffit de le réduire à un problème $K_2 \in \mathcal{C}$.

4.3 Réduction et complétude

Définition 4.6 *Soit $\mathcal{C}$ une classe de complexité et K un langage.*
– *K est $\mathcal{C}$-dur si tout langage de $\mathcal{C}$ se réduit à K.*
– *K est $\mathcal{C}$-complet si $K \in \mathcal{C}$ et K est $\mathcal{C}$-dur.*

Remarques :
– Si K_1 est $\mathcal{C}$-dur et K_1 se réduit à K_2 alors K_2 est $\mathcal{C}$-dur.
– Si K est $\mathcal{C}$-dur et $K \in \mathcal{C}'$ où $\mathcal{C}'$ est fermée par réduction, alors $\mathcal{C} \subseteq \mathcal{C}'$.

4.4 Formules booléennes

Formules booléennes : $\varphi ::= x \ (x \in X) \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi$
où $X = \{x_1, x_2, \ldots\}$ est un ensemble de variables booléennes.

Exemple : $\varphi = (x_1 \vee (\neg x_2 \wedge x_3)) \wedge \neg(\neg x_1 \vee x_2)$

Forme normale conjonctive (FNC).
Littéral : variable ou négation d'une variable : x ou $\neg x$.
Clause : disjonction de littéraux : $x_1 \vee \neg x_2 \vee x_3$.
FNC : conjonction de clauses : $(x_4 \vee \neg x_2) \wedge (x_1 \vee x_3) \wedge (x_2 \vee \neg x_3)$.

Représentation d'une formule sur l'alphabet $\Sigma = \{x, 0, 1, \neg, \vee, \wedge, (,)\}$:

$$(x100 \vee \neg x10) \wedge (x1 \vee x11) \wedge (x10 \vee \neg x11)$$

4.4 Assignations

Assignation : $\sigma : X \to \{0, 1\}$.

Évaluation : $\varphi(\sigma) \in \{0, 1\}$ (définition inductive sur φ).

Satisfaction : $\sigma \models \varphi$ si $\varphi(\sigma) = 1$.

Représentation d'une assignation : mot sur l'alphabet $\{0, 1\}$.

Exemple : $\sigma = 01101$.

4.4 Problème SAT

Problème SAT-VALUE : Évaluation d'une formule booléenne en FNC.
Donnée : une formule φ en FNC et une assignation σ.
Question : $\sigma \models \varphi$?

Proposition 4.7 $\mathrm{SAT{-}VALUE} \in \mathbf{TIME}(n^2) \cap \mathbf{SPACE}(\log n)$.

Problème SAT : Satisfaisabilité d'une formule booléenne en FNC.
Donnée : une formule φ en FNC.
Question : φ est-elle satisfaisable ?

Proposition 4.8
- $\mathrm{SAT} \in \mathbf{NP}$.
- $\mathrm{SAT} \in \mathbf{EXP}$.

4.4 Chemin Hamiltonien

Problème du chemin Hamiltonien :
Donnée : un graphe $G = (S, A)$.
Question : existe-t-il un chemin dans G qui passe une fois et une seule par chaque sommet ?

Proposition 4.9
– *Le problème du chemin Hamiltonien est dans* **NP**.
– *Le problème du chemin Hamiltonien se réduit au problème SAT.*

Un chemin Hamiltonien est une permutation $\pi(1), \ldots, \pi(n)$ telle que $\forall 1 \leq i < n$, $(\pi(i), \pi(i+1)) \in A$.
On le code par n^2 variables booléennes : $x_{i,j} = (\pi(i) = j)$.
On écrit une formule booléenne qui est vraie si l'assignation définit un chemin Hamiltonien.

4.4 Clauses de Horn

Une clause de Horn est une clause qui contient au plus un littéral positif.

Remarque : une clause de Horn est une implication :

$$\begin{array}{rcccl} \neg x_1 \vee x_2 \vee \neg x_3 & = & x_1 \wedge x_3 & \rightarrow & x_2 \\ x_1 & = & \mathrm{V} & \rightarrow & x_1 \\ \neg x_1 \vee \neg x_2 & = & x_1 \wedge x_2 & \rightarrow & \mathrm{F} \end{array}$$

Problème Horn-SAT :
Donnée : une formule φ en FNC n'utilisant que des clauses de Horn.
Question : φ est-elle satisfaisable ?

Proposition 4.10 $\mathrm{Horn{-}SAT} \in \mathbf{P}$.

4.5 Circuits booléens

Définition 4.11 *Un circuit booléen est un graphe acyclique étiqueté* $C = (S, A, \lambda)$ *où* $S = \{1, \ldots, m\}$ *est l'ensemble des portes du circuit,* $\lambda : S \to \{\mathrm{V}, \mathrm{F}, \wedge, \vee, \neg, x_1, x_2, \ldots, x_n\}$ *donne le type d'une porte, et vérifie :*

- $\lambda(k) \in \{\mathrm{V}, \mathrm{F}, x_1, x_2, \ldots, x_n\}$ *ssi* k *n'a pas d'arête entrante (porte d'entrée du circuit),*
- $\lambda(k) = \neg$ *ssi* k *a exactement une arête entrante,*
- $\lambda(k) \in \{\vee, \wedge\}$ *ssi* k *a exactement deux arêtes entrantes.*

On appelle porte de sortie une porte sans arête sortante.

Puisqu'un circuit est acyclique, il a au moins une porte de sortie (m).

Exemple : additionneur binaire avec retenue.

4.5 Représentation d'un circuit

Représentation d'un circuit : $\#c(1)\#c(2)\#\cdots\#c(m)\#$
sur l'alphabet $\Sigma = \{\mathrm{V}, \mathrm{F}, \neg, \vee, \wedge, x, 0, 1, \#\}$,
où $c(i)$ est la représentation de la porte i :

– si $\lambda(i) \in \{\mathrm{V}, \mathrm{F}\}$ alors $c(i) = \lambda(i)$,
– si $\lambda(i) = x_k$ alors $c(i) = x\,\mathrm{bin}(k)$,
– si $\lambda(i) = \neg$ alors $c(i) = \neg\,\mathrm{bin}(k)$, où k est la porte entrante de i,
– si $\lambda(i) = \vee$ alors $c(i) = \mathrm{bin}(k) \vee \mathrm{bin}(\ell)$, où k et ℓ sont les portes entrantes de i,
– si $\lambda(i) = \wedge$ alors $c(i) = \mathrm{bin}(k) \wedge \mathrm{bin}(\ell)$, où k et ℓ sont les portes entrantes de i

Note $\mathrm{bin}(k)$ est la représentation binaire de l'entier k.

Remarque : $|C| = \mathcal{O}(m \log(m))$.

4.5 Évaluation d'un circuit

Soit σ une assignation des variables du circuit ($\sigma \in \{\mathrm{V}, \mathrm{F}\}^*$).

On note $C(\sigma, k) \in \{\mathrm{V}, \mathrm{F}\}$ la valeur de la porte k lorsque les portes d'entrées variables du circuit sont définies par σ.

Problème CIRCUIT-VALUE : Évaluation d'un circuit booléen.
Donnée : un circuit booléen C, une assignation σ, une porte k.
Résultat : $C(\sigma, k)$.

Proposition 4.12 *CIRCUIT-VALUE se résout en temps quadratique.*

4.5 CIRCUIT-SAT

Problème CIRCUIT-SAT : Satisfaisabilité d'un circuit booléen.

Donnée : un circuit booléen C, une porte k.

Question : existe-t-il une assignation σ telle que $C(\sigma, k) = \mathrm{V}$?

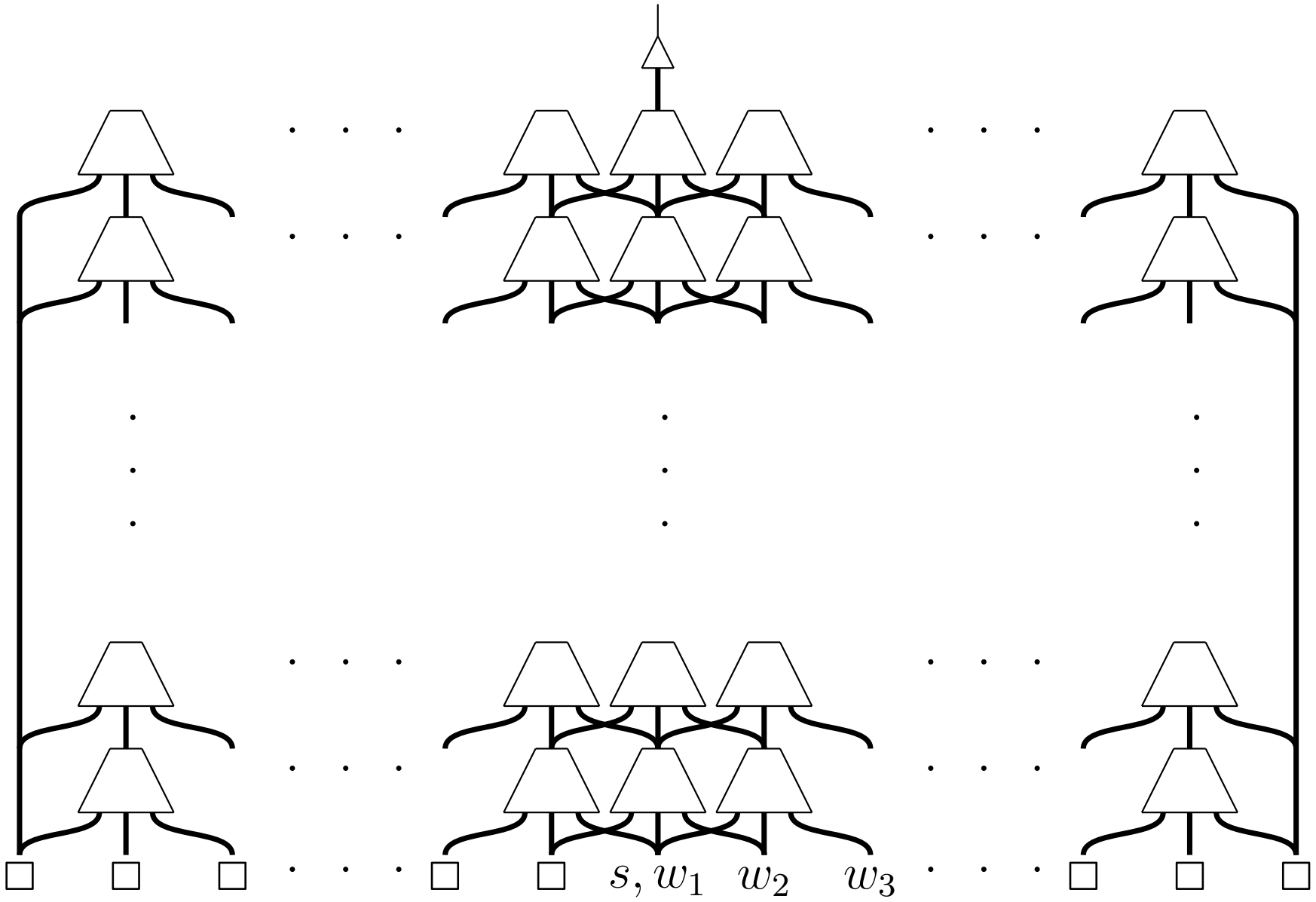
Proposition 4.13

- *CIRCUIT-SAT est dans* **NP**.
- *CIRCUIT-SAT se réduit à 3-SAT.*

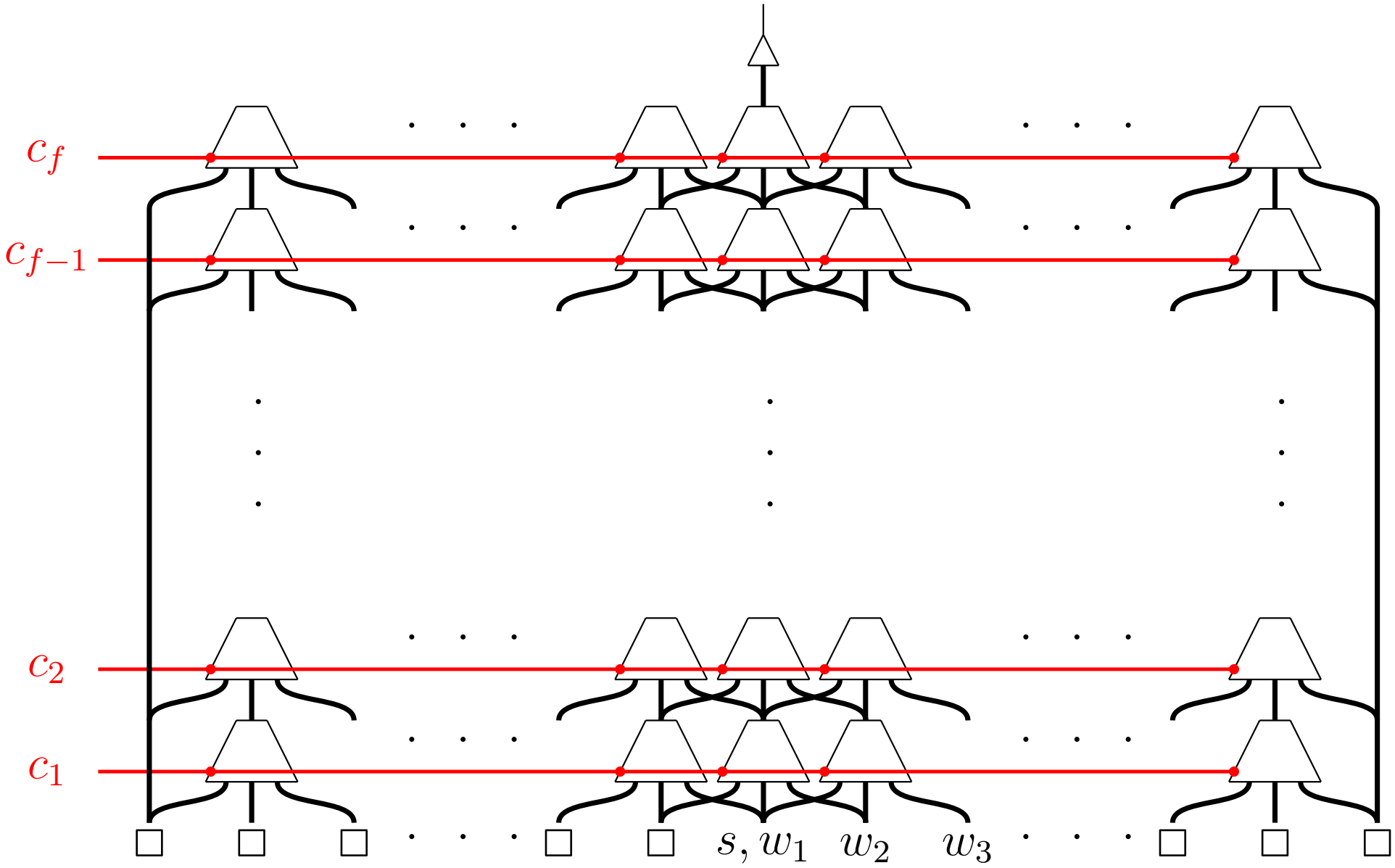
Théorème 4.14

- *CIRCUIT-VALUE est* **P***-complet (Ladner, 1975).*
- *CIRCUIT-SAT est* **NP***-complet.*
- *SAT est* **NP***-complet, (Cook, 1971).*

4.5 Évaluation d'un circuit



4.5 CIRCUIT-SAT



4.6 Relations entre classes de complexité

Définition 4.15 *$f : \mathbb{N} \to \mathbb{N}$ est une bonne fonction de complexité si f est croissante, et, en utilisant un codage unaire des entiers, f est calculable par une MT déterministe en temps $\mathcal{O}(n + f(n))$ et en espace $\mathcal{O}(f(n))$.*

Exemples :

- $n \mapsto c$ (fonction constante)
- $n \mapsto n$
- $n \mapsto \log n$
- $n \mapsto \sqrt{n}$

Proposition 4.16 *Si f et g sont des bonnes fonctions de complexité, alors $f + g$, $f \cdot g$, 2^f aussi.*

4.6 Relation espace/temps

Proposition 4.17 *Soit f une bonne fonction de complexité.*

$$\textbf{NTIME}(f) \subseteq \textbf{SPACE}(f)$$

Preuve: Soit M une MT (D ou ND) qui travaille en temps f.
On construit une MT déterministe M' qui sur une donnée w de M,

- calcule $f(|w|)$,
- énumère les suites de choix de longueur $f(|w|)$,
- simule le calcul de M pour chaque suite de choix. □

4.6 Graphe des configurations

Soit M une MT (D ou ND) qui travaille en espace f.
Soit w une donnée de M.
On note $G(M, w)$ le graphe des configurations de M sur une donnée w.

• Un sommet de $G(M, w)$ est une configuration de M sur w.

$$c = (p, i, (u_1, v_1), \ldots, (u_k, v_k))$$

où p est l'état, i la position de la tête sur la bande d'entrée,
(u_j, v_j) définit le contenu et la position de la tête sur la bande j.
Nombre de sommets de $G(M, w)$:

$$N = |Q| \cdot |w| \cdot \left(f(|w|) \cdot |\Gamma|^{f(|w|)}\right)^k = |Q| \cdot 2^{\mathcal{O}(\log |w| + f(|w|))}$$

• (c, c') est une arête de $G(M, w)$ s'il existe une transition de M qui fait passer de la configuration c à la configuration c'.

• c_0 : configuration initiale, F : configurations acceptantes.

Proposition 4.18 $w \in \mathcal{L}(M)$ *ssi* $\mathrm{Acc}(c_0) \cap F \neq \emptyset$ *dans* $G(M, w)$.

4.6 Relation temps/espace

Proposition 4.19 *Soit f une bonne fonction de complexité.*

$$\textbf{NSPACE}(f(n)) \subseteq \textbf{TIME}(2^{\mathcal{O}(\log n + f(n))})$$

Preuve: Soit M une MT (D ou ND) qui travaille en espace f.
On construit une MT déterministe M' qui sur une donnée w de M, teste si $\mathrm{Acc}(c_0) \cap F \neq \emptyset$ dans $G(M, w)$. □

Rem : Il n'est pas nécessaire de construire la représentation de $G(M, w)$.
On sait générer les configurations successeurs c' d'une configuration c.

4.6 Théorème de Savitch

Théorème 4.20 (Savitch)
L'accessibilité est décidable en **SPACE**$(\log^2 n)$.

Preuve: On considère le prédicat :
$\mathrm{Path}(x, y, i)$: il existe un chemin de x à y de longueur au plus 2^i.

$$\mathrm{Path}(x, y, i) = \bigvee_{z \in S} \mathrm{Path}(x, z, i-1) \wedge \mathrm{Path}(z, y, i-1)$$

La pile de récursivité nécessaire est donc de taille $\mathcal{O}(\log^2 n)$. □

4.6 Espace non déterministe

Théorème 4.21 *Soit f une bonne fonction de complexité telle que $\log n = \mathcal{O}(f(n))$. On a*

$$\textbf{NSPACE}(f) \subseteq \textbf{SPACE}(f^2)$$

Preuve: Soit M une MT (D ou ND) qui travaille en espace f.
On construit une MT déterministe M' qui sur une donnée w de M, teste en utilisant la MT de Savitch si $\mathrm{Acc}(c_0) \cap F \neq \emptyset$ dans $G(M, w)$.
Il est essentiel de ne pas générer $G(M, w)$ sur une bande auxiliaire. □

Corollaire 4.22 **PSPACE** = **NPSPACE**

4.6 Bilan

$$\mathbf{L} \subseteq \mathbf{NL} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} = \mathbf{NPSPACE} \subseteq \mathbf{EXP} \subseteq \mathbf{NEXP}$$

4.6 Calcul non déterministe d'une fonction

Une MT non déterministe calcule une fonction $h : \Sigma_1^* \rightarrow \Sigma_2^*$ en temps f et en espace g si pour tout $w \in \Sigma_1^*$

- tous les calculs de M sur w s'arrêtent après au plus $f(|w|)$ transitions et en utilisant un espace de travail au plus $g(|w|)$,
- au moins un calcul de M sur w s'arrête dans l'état OK,
- chaque calcul de M sur w qui s'arrête dans l'état OK contient $h(w)$ sur la bande de sortie à la fin du calcul.

Une MT ND peut abandonner, mais elle ne peut pas fournir une mauvaise réponse.

Remarque : calculer de façon ND la fonction caractéristique d'un langage revient à décider de façon ND le langage et son complémentaire.

4.6 Immerman – Szelepscényi

Problème $|\mathrm{Acc}|$: nombre d'accessibles
Données : Un graphe $G = (S, A)$ et un sommet $u \in S$.
G est représenté par $n = |S|$ en binaire et la matrice d'adjacence A,
$u \in S = \{1, \ldots, n\}$ est donné en binaire.
Question : Calculer $|\mathrm{Acc}(u)|$.

Théorème 4.23 (Immerman – Szelepscényi)
Le calcul du nombre d'accessibles est dans **NL**.

Il est facile de décider dans **NL** si $|\mathrm{Acc}(u)| \geq K$.
Il est moins facile de décider dans **NL** si $|\mathrm{Acc}(u)| < K$.

On note $S(k)$ l'ensemble des sommets accessibles à partir de u par un chemin de longueur au plus k.

4.6 Immerman – Szelepscényi

Données : $n = |S|$, A, $u \in S$

Problème : calculer $|\mathrm{Acc}(u)| = |S(n-1)|$.

Machine M_0. Variables auxiliaires : $v \in S$ et $k, \ell, m \leq n$

$m := 1$ $\quad\quad$ $m = |S(0)|$

Pour $k := 1$ à $n-1$ faire $\quad\quad$ $m = |S(k-1)|$: invariant

  $\ell := 0$

  Pour $v := 1$ à n faire

    Si M_1 (i.e. $v \in S(k)$) alors $\ell := \ell + 1$

  FPour

  $m := \ell$

FPour

Écrire m sur la bande de sortie; STOP(OK)

M_0 calcule $|\mathrm{Acc}(u)| = |S(n-1)|$.

Si M_1 s'arrête dans l'état OK, la réponse est correcte.

4.6 Immerman – Szelepscényi

Données supplémentaires : $v \in S$, $k \leq n$, $m \leq n$

Problème : tester si $v \in S(k)$ connaissant $m = |S(k-1)|$.

Machine M_1. Variables auxiliaires : $w \in S$, $q \leq n$

$q := 0$

Pour $w := 1$ à n faire

  Si M_2 (i.e. $w \in S(k-1)$) alors

    $q := q + 1$

    Si $v = w$ ou $(w, v) \in A$ alors STOP(V)

  fsi

FPour

Si $q = m$ alors STOP(F) sinon STOP(KO) fsi

M_1 calcule $v \in S(k)$ si $m = |S(k-1)|$.

Si M_1 répond V ou F, la réponse est correcte.

M_1 abandonne (STOP(KO)) uniquement si M_2 a fait une erreur.

4.6 Immerman – Szelepscényi

Données supplémentaires : $w \in S$, $k \leq n$
Problème : tester si $w \in S(k-1)$.

Machine M_2. Variables auxiliaires : $x, y \in S$, $p \leq n$
$x := u$
Pour $p := 1$ à $k-1$ faire
  Soit $y \in \{1, \ldots, n\}$ (non déterminisme)
  Si $y = x$ ou $(x, y) \in A$ alors $x := y$ sinon STOP(F) fsi
FPour
Si $x = w$ alors STOP(V) sinon STOP(F) fsi

M_2 ne calcule pas $w \in S(k-1)$.
Si M_2 répond V alors $w \in S(k-1)$.
Si M_2 répond F, on ne sait rien.
M_2 n'abandonne jamais.

4.6 non accessibilité

Problème $\overline{\text{Acc}}$: non accessibilité dans un graphe

Données : Un graphe $G = (S, A)$ et deux sommets $u, v \in S$.

G est représenté par $n = |S|$ en binaire et la matrice d'adjacence A,

$u, v \in S = \{1, \ldots, n\}$ sont donnés en binaire.

Question : Est-ce que v n'est pas accessible à partir de u dans G ?

Corollaire 4.24 $\overline{\text{Acc}} \in$ **NL**

4.6 non accessibilité

Données : $n = |S|$, A, $u, v \in S$

Problème : décider si $v \notin \mathrm{Acc}(u)$.

Machine M_3. Variables auxiliaires : $w \in S$ et $k, m \leq n$

$m := |\mathrm{Acc}(u)|$ (avec la machine M_0)

$k := 0$

Pour $w := 1$ à n faire

  Si M_4 (i.e. $w \in \mathrm{Acc}(u)$) alors

    $k := k + 1$

    Si $w = v$ alors STOP(F) Fsi

  Fsi

FPour

Si $m = k$ alors STOP(V) sinon STOP(KO) fsi

Si M_3 répond V ou F alors la réponse est correcte.

M_3 abandonne (STOP(KO)) uniquement si M_4 a fait une erreur.

4.6 non accessibilité

Donnée supplémentaire : $w \in S$

Problème : tester si $w \in \mathrm{Acc}(u)$.

Machine M_4. Variables auxiliaires : $x, y \in S$, $p \leq n$

$x := u$

Pour $p := 1$ à $n - 1$ faire

  Soit $y \in \{1, \ldots, n\}$ (non déterminisme)

  Si $y = x$ ou $(x, y) \in A$ alors $x := y$ sinon STOP(F) fsi

FPour

Si $x = w$ alors STOP(V) sinon STOP(F) fsi

M_4 ne calcule pas $w \in \mathrm{Acc}(u)$.

Si M_4 répond V alors $w \in \mathrm{Acc}(u)$.

Si M_4 répond F, on ne sait rien.

M_4 n'abandonne jamais.

4.6 Complémentaires

Si $\mathcal{C}$ est une classe de complexité, alors **co-**$\mathcal{C} = \{L \mid \overline{L} \in \mathcal{C}\}$.

Proposition 4.25 *Si $\mathcal{C}$ est une classe déterministe alors* **co-**$\mathcal{C} = \mathcal{C}$.

Théorème 4.26 *Soit f une bonne fonction de complexité telle que $f(n) \geq \log n$. On a*

$$\textbf{NSPACE}(f) = \textbf{co-NSPACE}(f)$$

Preuve: On utilise le graphe des configuration et $\overline{\mathrm{Acc}} \in$ **NL**. □

Corollaire 4.27 **NL** = **co-NL**