Classes de complexité parallèles

Langage calculé par une famille de ci

- Soit $C = (C_n)_{n \in \mathbb{N}}$ une famille de circuits, tel que C_n à n-entrées. C calcule le langage $L \subset \{0,1\}^*$ si $\forall x \in \{0,1\}^*, |x| = n, x \in L \Leftrightarrow C_n(x) = 1$.
- La **taille** de C est le nombre de portes de C.
- La **profondeur** de C est la longueur du plus long chemin entre une entrée et une sortie.

Complexité de circuits

- Soit $f : \mathbb{N} \to \mathbb{N}$ une fonction. Un langage $L \subset \{0,1\}*$ est dans $CIRCUIT(f)$ s'il existe une famille de circuits $C = (C_n)_{n \in \mathbb{N}}$ qui calcule L avec $taille(C_n) = O(f(n))$.
- **Théorème:** si $L \in TIME(f(n))$ alors $L \in CIRCUIT(f^2(n))$.

  Démonstration: Etant donné n, on peut simuler le calcul d'une machine ML sur les entrées de taille n par un circuit de taille $O(f^2(n))$ en considérant un circuit qui simule le calcul de M sur une entrée de taille n, en utilisant exactement la même technique que pour le circuit construit dans la démonstration de la P-complètude de CIRCUIT-VALUE.

Conséquences

- **Corollaire:** Les langages de P admettent des circuits de taille polynômiale.

P et circuits polynomiaux non-uniform

- **Question:** Est-ce que les circuits de tailles polynomiaux correspondent à P.
- **Réponse:** Non. Il existe une famille de circuits de taille polynomiale qui reconnait un problème indécidable.

Démonstration. Considèrons un langage $L \subset \{1^n | n \in \mathbb{N}\}$ indécidable. (Par exemple, $L = \{1^n |$ la $n$ème machne de Turing s'arrête$\}$).
$L' = \{x | 1^{|x|} \in L\}$ est indécidable: si il y avait un algorithme pour le résoudre, il y aurait clairement un algorithme pour L.
Pourtant L' est reconnu par la famille de circuits $(C_n)_{n \in \mathbb{N}}$ définie par:

1. **C_n est le circuit constant qui vaut 1 si $1^n \in L$.**
2. **C_n est le circuit constant qui vaut 0 sinon.**

P et circuits polynomiaux uniformes

- On fixe un codage des circuits sur l'alphabet $\{0,1\}$ considéré.
- **Définition:** Une famille de circuits est L-uniforme (uniforme) si la fonction f qui à 1^n associe la description de C_n est calculable en espace $O(\log n)$.
- **Théorème:** $L \in P$ ssi L est accepté par une famille de circuits L-uniforme.

  Démonstration: $\Rightarrow$: Il suffit d'observer que dans la démonstration du théorème précédent, on peut bien, étant donné n en unaire, calculer automatiquement le circuit C_n en utilisant un espace $O(\log n)$.

  $\Leftarrow$: Pour déterminer si $x \in L$, calculer $n = |x|$, puis C_n, puis $C_n(x)$ et accepter si et seulement si $C_n(x) = 1$.

Conséquences

- Les familles de circuits *uniformes* de taille polynomiale correspondent au temps polynomial.

- Autrement dit, la taille d'une famille de circuit *uniforme* correspond au temps de calcul séquentiel.

Conséquences

- La profondeur d'une famille de circuit correspond elle au temps parallèle: tous les portes d'un circuit à même distance des entrées peuvent se calculer en parallèle, et donc la plus longue distance entre une entrée et une sortie (= la profondeur) correspond au temps qu'il faut pour évaluer le circuit en parallèle.

- $\rightsquigarrow$ On va définir les classes parallèles en considérant des restrictions sur la profondeur des familles de circuit.

Classes NC^i et AC^i.

- Définition: $NC^i = \{L \in \{0,1\}^*, L$ est reconnu par une famille de circuits uniforme de taille polynomiale et de profondeur $O((\log n)^i)\}$.
- Définition: AC^i comme NC^i mais on autorise les portes $\vee$ et $\wedge$ à avoir un nombre quelquonque d'entrées.
- **Théorème:** Pour tout i,
  1. $NC^i \subset AC^i$
  2. $AC^i \subset NC^{i+1}$

  Démonstration: 1. évident. 2. remplacer les portes $\vee$ et $\wedge$ à k-arguments par un arbre binaire de portes binaires de profondeur $\lceil \log k \rceil$.

Classe NC

- **Définition:**

$$NC = \bigcup_i NC^i.$$

- **Corollaire:** $NC \subset P$.
- **Corollaire:** $AC^0 \subset NC^1 \subset AC^1 \subset \ldots \subset P$.

Classes parallèles et espace logarithm

- **Théorème:** $NC^k \subset SPACE(\log^k n)$ pour tout k.

Démonstration: il suffit de montrer qu'on peut évaluer un circuit de profondeur $\log^k n$ **en espace** $O(\log^k n)$**. Principe évaluer le circuit par un parcours de graphe en profondeur en commençant par la sortie. Problème: on ne peut pas garder sur la pile de la récurrence les noms des sommets visités parceque cela ferait un espace total** $O(\log^{k+1} n)$**. Solution: en fait, on a uniquement besoin de stocker le numéro du noeud courant** g**, la porte logique correspondante, et les numéros de ses pères** g_d **et** g_r**, et de mémoriser à chaque niveau de la récurrence, un nombre constant de bits qui indique pour les sommets concernés, par exemple que le sous-arbre gauche a déja été évalué, qu'il vaut** 0 **(ou** 1**), et qu'on est en train d'évaluer le sous-arbre droit. Pour accèder à l'un des pères** g_d **et** g_r **du noeud courant** g**, la condition d'uniformité du circuit nous garanti que l'on peut bien mettre à jour** g, g_d **et** g_r **en utilisant un espace** $\log n$**.**

Classes parallèles et espace logarithm

- **Théorème:** $NC^1 \subset L \subset NL \subset AC^1$.

- **Démonstration $NL \subset AC^1$. Puisque $REACH$ est NL-complet, il suffit de montrer que $REACH \in AC^1$. Soit $G = (V, E)$ un graphe et M sa matrice adjacence: $M_{i,j} = 1$ ssi $(i, j) \in E$. Considèrons M^k définie par $(M^k)_{i,j} = 1$ ssi il existe un chemin de longueur k entre i et j. Chaque M^i peut se calculer par un circuit de profondeur $logi$ en utilisant récursivement l'identité $M^{2j} = M^j \times M^j$: le produit est ici défini par $(A \times B)_{i,j} = \vee_k (A_{i,k} \wedge B_{k,j})$. $REACH$ correspond alors $REACH = M^1 \vee M^2 \vee \ldots \vee M^n$.**

- **Démonstration $NC^1 \subset L$: c'est le théorème précédent pour $k = 1$.**

Exemples de problèmes de NC

- Problème ***ADDITION*:**
  - **Instance:** a,b,c en binaire
  - **Question:** $c = a + b$?

  est dans AC^0.

  Démonstration: Ecrit $a = a_{n-1}a_{n-2}\ldots a_0$, $b = b_{n-1}b_{n-2}\ldots b_0$,
  $c = c_{n-1}c_{n-2}\ldots c_0$. $c_0 = a_0 \wedge b_0$. $c_i = (a_i \wedge b_i) \vee (r_{i-1} \wedge (a_i \vee b_i))$. $g_i = a_i \wedge b_i$.
  $p_i = a_i \vee b_i$. $r_i = g_i \vee (g_{i-1} \wedge p_i) \vee (g_{i-2} \wedge p_{i-1} \wedge p_i) \wedge (g_0 \wedge p_1 \wedge \ldots \wedge p_i)$.
  g_i, p_i **profondeur** 1. r_i **profondeur** 3. c_i **profondeur** 5.

Exemples de problèmes de NC

- Problème ***ITERATEDADDITION*:**
  - **Instance:** $a_1, \ldots, a_n, s$ en binaire
  - **Question:** $s = \sum_i a_i$?

  est dans NC^1.

  Démonstration. Truc 3POUR2: l'addition de 3 nombres se réduit à l'addition de 2 nombre: supposons $a = a_{n-1}a_{n-2}\ldots a_0$**,** $b = b_{n-1}b_{n-2}\ldots b_0$**,** $c = c_{n-1}c_{n-2}\ldots c_0$**.** $a_i + b_I + c_i$ **s'écrit en binaire** $p_i q_i$**. Considèrons** $p = p_{n-1}p_{n-2}\ldots p_0$ **et** $q = q_{n-1}q_{n-2}\ldots q_0$**. On a alors** $a + b + c = 2p + q$**.**

  On répète le truc $\log n$ **fois.**

Exemples de problèmes: suite

- Problème ***MULTIPLICATION***:
  - **Instance:** a,b,c en binaire
  - **Question:** $c = a * b$?

  est dans NC^1.

  Démonstration: écrire multiplication comme addition itèrée sur décalages.
- Problème ***INTEGER MATRIX MULTIPLICATION***:
  - **Instance:** A,B,C matrices en binaire
  - **Question:** $C = A * B$?

  est dans NC^1.

  Démonstration: porte (i, j, k) **calcule** $a_{i,k}b_{k,j}$ **en** NC^1**. calcule ensuite** $c_{i,j} = \sum_k a_{i,k}b_{k,j}$ **en** NC^1**.**

Exemples de problèmes: suite

- Problème ***ITERADED INTEGER MATRIX MULTIPLICATION*:**
  - **Instance:** $A_1, \ldots, A_n, C$ matrices en binaire
  - **Question:** $C = A_1 \times \ldots \times A_n$?

  est dans NC^2.

  Démonstration: construit un circuit correspondant à un arbre binaire de hauteur $\log n$ dont chaque noeud correspond à un circuit de profondeur $\log n$.

Exemples de problèmes: suite

- Problème ***POLYNOME CARACTERISTIQUE*:**
  - **Instance:** A une matrice, P un polynome
  - **Question:** $P = det(A - \lambda I)$?

  est dans NC^2.

  Démo: exercice.*
- **Corollaire:** Determinant est dans NC^2.

  Démonstration: le déterminant est donné par le dernier coefficient du polynome caractéristique.

Bornes inférieures: Problème PARIT

- **Théorème:** Problème ***PARITE*:**
  - **Instance:** $a_1, \ldots, a_n \in \{0,1\}$,$p$
  - **Question:** $p = \sum_{a_i} mod 2$?

  est dans NC^1 mais n'est pas dans AC^0.

  Démo: exercice.*

Réductions

- Une fonction g est AC^0 réductible à f s'il existe un circuit de taille polynomial de profondeur constante avec des portes $\neg, \vee, \wedge$ et des portes f qui calcule g.
- **Définition:** $TH_k(X_1, \ldots, X_n)$ vaut 1 si $\sum_i X_i \geq k$, 0 sinon.
- **Définition:** $TRI(X_1, \ldots, X_n) = (TH_n(X_1, \ldots, X_n), \ldots, Th_1(X_1, \ldots, X_n))$.
- **Théorème:** $TRI \in NC^1$. $TRI \notin AC^0$.

  Démonstration. TRI est clairement dans NC^1.

  $PARITE(x_1, \ldots, x_n) = \vee_{i\ impair} TH_i(X) \wedge \neg TH_{i+1}(X)$ **et donc PARITE se réduit à TRI.**

- **Définition:**
  $MAJORITE(X_1, \ldots, X_n) = TH_{n/2}(X_1, \ldots, X_n)$.
- **Théorème:** $MAJORITE \notin AC^0$.

  Démonstration. Pour tout k, $TH_k(X_1, \ldots, Xn)$ se réduit à $MAJORITE$ par définition.