Complexity Theory

Jan Křetínský

Chair for Foundations of Software Reliability
and Theoretical Computer Science
Technical University of Munich

Summer 2016

Based on slides by Jörg Kreiker

Lecture 23

NC and AC scrutinized

Recap

Efficient parallel computation

- computable by some PRAM with
- polynomially many processors in
- polylogarithmic time
- robust wrt to underlying PRAM model

Recap

Efficient parallel computation

- computable by some PRAM with
- polynomially many processors in
- polylogarithmic time
- robust wrt to underlying PRAM model

corresponds to

small depth circuits

- of polynomial size
- polylogarithmic depth
- logspace uniform

Recap – NC and AC

If $L \subseteq \{0,1\}^*$ is decided by a logspace-uniform family $\{C_n\}$ of polynomially sized circuits with bounded fan-in

- and depth $\log^k n$ then $L \in \mathbf{NC^k}$ for $k \geq 0$
- $\mathbf{NC} = \bigcup_{k \geq 0} \mathbf{NC^k}$

Recap – NC and AC

If $L \subseteq \{0,1\}^*$ is decided by a logspace-uniform family $\{C_n\}$ of polynomially sized circuits with bounded fan-in

- and depth $\log^k n$ then $L \in \mathbf{NC^k}$ for $k \geq 0$
- $\mathbf{NC} = \bigcup_{k \geq 0} \mathbf{NC^k}$

If the fan-in is unbounded we obtain the corresponding **AC** hierarchy.

Goal

Find the places of **NC** and **AC** among other complexity classes!

Agenda

- **NC** versus **AC**
- **NC** versus **P**
- $\mathbf{NC^1}$ versus **L**
- $\mathbf{NC^2}$ versus **NL**

Unbounded → bounded fan-in

Theorem

For all $k \geq 0$

$$\mathsf{NC}^k \subseteq \mathsf{AC}^k \subseteq \mathsf{NC}^{k+1}$$

Unbounded → bounded fan-in

Theorem

For all $k \geq 0$

$$\mathbf{NC}^{k} \subseteq \mathbf{AC}^{k} \subseteq \mathbf{NC}^{k+1}$$

Proof

- first inclusion trivial
- for the second, assume $L \in \mathbf{AC}^{k}$ by family $\{C_n\}$
- there exists a polynomial $p(n)$ such that
  - C_n has $p(n)$ gates with
  - maximal fan-in of at most $p(n)$
- each such gate can be simulated by a binary tree of gates of the same kind with depth $\log(p(n)) = \mathcal{O}(\log n)$

⇒ the resulting circuit has size at most size $p(n)^2$, depth at most $\log^{k+1} n$ and maximal fan-in 2

Corollary

Theorem

AC $=$ NC

Corollary

Theorem

AC $=$ NC

Remarks

- the inclusions in the theorem on the previous slide are strict for $k = 0$
- one strict inclusion is trivial, the other one is subject of the next lecture
- for practical relevance, we focus on bounded fan-in, ie. **NC**

Agenda

- **NC** versus **AC** ✓
- **NC** versus **P**
- $\mathbf{NC}^1$ versus **L**
- $\mathbf{NC}^2$ versus **NL**

NC versus P

Theorem

NC $\subseteq$ **P**

Proof

- let $L \in$ **NC** by circuit family $\{C_n\}$

$\Rightarrow$ there exists a logspace TM M that computes $M(1^n) = desc(C_n)$

- the following **P** machine decides L
  - on input $x \in \{0,1\}^n$ simulate M to obtain $desc(C_n)$
  - C_n has input variables $z_1, \ldots, z_n$
  - evaluate C_n under the assignment σ that maps z_i to the $i - th$ bit of x
  - output $C_n(\sigma)$
- all steps take polynomial time (evaluation takes time proportional to circuit size)

Remarks

- **P** equals the set of languages with logspace-uniform circuits of polynomial size and polynomial depth (exercise)
- it is an open problem whether the previous inclusion is strict
- in fact it is open whether $\mathbf{NC}^1 \subset \mathbf{PH}$
- problem is important, since it answers whether all problems in **P** have fast parallel algorithms
- conjecture: strict

Agenda

- **NC** versus **AC** ✓
- **NC** versus **P** ✓
- $\mathbf{NC}^1$ versus **L**
- $\mathbf{NC}^2$ versus **NL**

Proof Steps

1. logspace reductions are transitive
2. if $L \in \mathbf{NC^1}$ then there exists a logspace uniform family of circuits $\{C_n\}$ of depth $\log n$
3. circuit evaluation of a circuit of depth d and bounded fan-in can be done in space $\mathcal{O}(d)$

What is the theorem?

What is the theorem?

Theorem

$\mathbf{NC}^1 \subseteq \mathbf{L}$.

Proof

- for a language $L \in \mathbf{NC}^1$, we can compute its circuits (step 2) in logspace
- we can evaluate circuits in logspace (step 3)
- the composition of these two algorithms is still logspace (step 1)
- steps 1 and 2 already proven

Proof of Step 3

- evaluate the circuit recursively
- identify gates with paths from output to input node
  - output node: ϵ
  - left predecessor of gate π: $\pi.0$
  - right predecessor of gate π: $\pi.1$

Proof of Step 3

- evaluate the circuit recursively
- identify gates with paths from output to input node
  - output node: ϵ
  - left predecessor of gate π: $\pi.0$
  - right predecessor of gate π: $\pi.1$
- 1. if π is an input return value
  2. if π denotes an op gate, compute value of $\pi.0$, value of $\pi.1$ and combine
- recursive depth $\log n$, only one global variable holding current path: total $\log n$ space
- note that the naive recursion takes $\log^2 n$ space!

Agenda

- **NC** versus **AC** ✓
- **NC** versus **P** ✓
- $\mathbf{NC}^1$ versus **L** ✓
- $\mathbf{NC}^2$ versus **NL**

The theorem

Theorem

$\mathbf{NL} \subseteq \mathbf{NC}^2$

Proof outline

- show that Path $\in$ $\mathbf{NC}^2$
- let $L \in \mathbf{NL}$ and $\mathbf{NL}$ machine M deciding it; for a given input $x \in \{0, 1\}^*$
- build a circuit C_1 computing the adjacency matrix of M's configuration graph on input x
- build a second circuit C_2 that takes this output and decides whether there is an accepting run
- the composition of C_1 and C_2 decides L
- observe: the composition can be computed in logspace

Path $\in$ **NC^2**

- let A be the $n \times n$ adjacency matrix of a graph
- let $B = A + I$ (add self loops)
- compute the square product B^2

$$B^2_{i,j} = \bigvee_k B_{i,k} \wedge B_{k,j}$$

- contains 1 iff there is a path of length at most 2
- can be done in **AC^0** $\subseteq$ **NC^1**
- $\log n$ times repeated squaring

$\Rightarrow$ paths can be computed in **NC^2**

Agenda

- **NC** versus **AC** ✓
- **NC** versus **P** ✓
- $\mathbf{NC}^1$ versus **L** ✓
- $\mathbf{NC}^2$ versus **NL** ✓

Criticism of NC

The notion of **NC** as efficient parallel computation may be criticized.

- polynomially many processors
  - in the **NC** hierarchy a $\log n$ algorithm with n^2 processors is favored over one with n processors and time $\log^2 n$
  - expensive
- polylogarithmic depth
  - for many practical inputs, sublinear algorithms might be as good or better
  - e.g. $n^{0.1}$ is at most $\log^2 n$ for values up to 2^{100}

Summary

- **AC** $=$ **NC**
- **NC**$^1 \subseteq$ **L** $\subseteq$ **NL** $\subseteq$ **NC**$^2 \subseteq$ **P**
- up next: **AC**$^0 \subset$ **NC**^{1}