

Computational Complexity Theory

Lecture notes by Henrik Hald Nørgaard

Lecture 14, 13/10 2006

1 Topics for the Second Part of the Course

There are 3 themes in modern complexity theory that we could consider:

1. Interactive and probabilistically checkable proofs (PCP-theorem)
2. Derandomization (towards $P = RP$ and $prP = prRP$ - the intuition is that these equalities hold. Proof of Reingolds result: $UGAP \in L$)
3. Constant depth computation

Other suggestions or preferences for one of the above are welcome...

The third theme will be the subject of this lecture. Constant depth computation is a classical topic that was studied in particular in the 1980'ties. It is the study of tasks that can be solved 'immediatly' using a few massively parallel computations. Examples of such tasks are rudimentary computational operations such as multiplication and addition of integers. There are some negative result in this field, e.g. stating that there are certain tasks that the human brain cannot solve immediatly.

2 ATM with Random Access

When studying constant depth computations the top of the hierarchy of complexity classes is L . This means that our previous reductions break down. In the rest of the lecture, we will formalize the meaning of constant depth computation and define the complexity classes that we will use to study this kind of computation.

We have previously defined a model for massive parallelism, namely alternating Turing machines. This model will not be suitable here. Instead we will study a related model for sublinear time massive parallelism.

Definition 1 (Alternating TM with random access) *An alternating Turing machine with random access is an alternating Turing machine with an output tape, that additionally differs from a the ATM from lecture 6 in the following way:*

1. The TM only gets the length in binary of the input x on the input tape
2. The TM will write an integer i and a bit b on the output tape and then halt
3. The TM accepts iff $x_i = b$

Running times for an ATM with random access are measured in terms of $|x|$ and thus the classes $ATS(\cdot, \cdot, \cdot)$ can be defined as for ordinary ATM's. The time component should though be sublinear for ATM's with random access. We now have two definitions of $ATS(\cdot, \cdot, \cdot)$ - and 'old' and a 'new' one. They are related in the following way:

$$ATS_{old}(\theta(a), t^{\theta(1)}, \theta(s)) = ATS_{new}(\theta(a), t^{\theta(1)}, \theta(s)) \quad (1)$$

for $t \geq n$ and $s \geq \log(n)$. We will not prove this fact.

3 Logspace Uniform Circuit Families

The circuit computed 'in the desert' by an ATM with random access only depends on the length of the input and thus defines a uniform circuit family C_1, C_2, \dots , that will define a language L . The ATM with random access and the corresponding circuit family are related in the following way (looking at inputs of a fixed length):

$$\text{Parallel time of machine} = \text{Depth of circuit} \quad (2)$$

$$\text{Parallel space (s) of machine} = \text{Size of circuit } (2^{\theta(s)}) \quad (3)$$

$$\text{Alternations of machine} = \text{Depth of the corresponding unbounded fan-in circuit} \quad (4)$$

The fan-in of a gate in a circuit is the number of inputs to the gate and the fan-in of a circuit is the maximal fan-in of a gate in the circuit. In the circuit corresponding to an ATM with random access, the gates can be assumed to be either AND- or OR-gates corresponding to the two kinds of states in the ATM (\neg -states can be eliminated). The gates of the circuit are naturally organized in layers, where two gates of the same type are in the same layer, if the corresponding states are entered in the same alternation of the ATM. Thus, the number of alternations of the ATM corresponds to the number of layers in the circuit. There may be wires between gates in the same layer, but each layer can be converted into a layer of gates whose wires are all to gates in different layers. This transition turns the circuit into a circuit of unbounded fan-in and for such a circuit the number of layers equals the depth of the circuit.

Using the above correspondence we will use an alternate point of view as our model for constant depth computation: We will look at circuit families with some uniformity condition. This leads us to the following definitions:

Definition 2 (Logspace uniformity) A circuit family $(C_i : \{0, 1\}^i \rightarrow \{0, 1\})_{i \geq 1}$ is called *logspace uniform* if there exists a logspace TM with output tape that on input 1^n outputs C_i .

Definition 3 (NC^i and AC^i)

$$NC^i = \{L \mid L \text{ is decided by a logspace uniform family of bounded} \quad (6)$$

$$\text{fan-in circuits of depth } O((\log n)^i) \text{ and polynomial size}\} \quad (7)$$

$$AC^i = \{L \mid L \text{ is decided by a logspace uniform family of unbounded} \quad (8)$$

$$\text{fan-in circuits of depth } O((\log n)^i) \text{ and polynomial size}\} \quad (9)$$

We also define $NC := \bigcup_{i \geq 1} NC^i$.

Example: $AND = \{1^n \mid n \geq 1\} \in NC^1$. To see this, build a binary tree with n leaves \cong input gates and inner nodes consisting of AND-gates. Also, $AND \in AC^0$. Here we can just use one (big) AND-gate of fan-in n .

Lemma 1

$$NC^i \subseteq AC^i \subseteq NC^{i+1} \subseteq P \quad (10)$$

Proof: The first inclusion holds by definition of NC^i and AC^i . To prove the second inclusion, notice that a gate in an AC^i -circuit can have at most $p(n)$ inputs as the number of wires is polynomially bounded. Such a gate can be converted to a tree of binary gates of depth $O(\log(n))$ and performing this conversion in the whole circuit gives a logarithmic increase in the depth of the circuit and a polynomial increase in the size. Thus the resulting circuit is in NC^{i+1} . Finally, a circuit of polynomial size can be evaluated in polynomial time and thus $NC \subseteq P$.

Remark: Notice that this above lemma implies that

$$NC = \bigcup_{i \geq 1} NC^i = \bigcup_{i \geq 1} AC^i \quad (11)$$

(and that is why we didn't define AC).

Remark: As a rule-of-thumb our definition of NC has less restrictive uniformity conditions than the circuits resulting from our definition of ATM with random access. A technical result is that

$$NC = AT_{new}(\cdot, (\log(n))^{O(1)}, O(\log(n))) \quad (12)$$

$$= AT_{old}((\log(n))^{O(1)}, n^{O(1)}, O(\log(n))) \quad (13)$$

Proposition 1

$$NC^1 \subseteq L \subseteq NL \subseteq AC^1 \quad (14)$$

Proof: $NC^1 \subseteq L$: Given input x construct the circuit $C_{|x|}$ in logspace. Then convert $C_{|x|}$ to a formula and evaluate it on input x . The last two steps can also be performed in logspace and thus we have a composition of 3 logspace computations, which is again a logspace computation.

$NL \subseteq AC^1$: It is enough to prove that the NL -complete problem GAP is in AC^1 . Let A be the adjacency matrix for the given graph G with 1's added on the diagonal and compute A^2, A^4, \dots, A^{2^k} , where $k = \lceil \log(n) \rceil$, n = number of nodes in the graph. The resulting matrix is the adjacency matrix for the transitive closure of the graph. Using this matrix is it trivial to solve GAP . As the adjacency matrix of G has entries in $\{0, 1\}$ the matrix multiplications can be modelled as a circuit using (\wedge, \vee) -matrix multiplication:

$$c_{ij} = \bigvee_{k=1}^n a_{ik} \wedge b_{kj} \quad (\text{Depth 2 circuit}) \quad (15)$$

The complete circuit has depth $O(\log(n))$.

Remark: The above algorithm for GAP results in the same circuit as using Savitch's algorithm from lecture 2.

4 Oracles and reductions

Definition 4 (Circuits using an oracle)

If A denotes an oracle then

$$NC^i[A] = \{L \mid L \text{ is decided by a logspace uniform family of bounded fan-in } (16)$$

$$A\text{-circuits of depth } O((\log n)^i) \text{ and polynomial size.} \quad (17)$$

$$\text{An } A\text{-gate is charged } \log(t) \text{ in terms of depth and } t \text{ in terms} \quad (18)$$

$$\text{of size, where } t \text{ is the number of inputs to the gate} \quad (19)$$

An A -gate is a special gate in the circuit where the inputs are interpreted as an input to the oracle and the output is likewise the answer from the oracle.

$$AC^i[A] = \{L \mid L \text{ is decided by a logspace uniform family of unbounded} \quad (20)$$

$$\text{fan-in } A\text{-circuits of depth } O((\log n)^i) \text{ and polynomial size.} \quad (21)$$

$$\text{An } A\text{-gate is charged } 1 \text{ in terms of depth and } t \text{ in terms} \quad (22)$$

$$\text{of size, where } t \text{ is the number of inputs to the gate} \quad (23)$$

AC^0 will replace P in the notion of reduction that we will use for constant depth computations:

Definition 5 (Reduction) *Let L_1 and L_2 be languages. Then L_1 reduces to L_2 under constant depth computations iff L_1 can be solved by an AC^0 -circuit using an L_2 -oracle:*

$$L_1 \leq_{cd} L_2 \Leftrightarrow L_1 \in AC^0[L_2] \quad (24)$$

4.1 Complete problems for L

Define PAP = Path Accessibility Problem to be GAP , but where the in and out degree of each vertex of the graph is at most 1 - thus reducing the graph to a union of paths and cycles. $PAP \in L$ as you can write a $O(1)$ -variable program that follows the edges from a starting node until it

1. finds the wanted node
2. can't go any further in the graph
3. times out

Another problem is FAP = Forest Accessibility Problem. This problem is also the same as GAP , but where the graph is a forest, that is: the in and out degree are at most 1 and the graph is acyclic. Also $FAP \in L$ using the same simple program as above. Now $FAP \leq PAP$: Use the Euler-path of the tree containing the start node.

Furthermore, FAP (and thus PAP) is complete for L : To see this consider the configuration graph of a language in L . As the TM deciding to this language is deterministic, the in and out degree of each node is at most 1. The graph can also be assumed to be acyclic by adding a timer to the configuration nodes and letting the TM halt when it has computed for so long that it must be in a loop. Now acceptance of the TM can be decided by using FAP on the modified configuration graph. This also shows that every L -computation can be made reversible.

A result from 2005 by Reingold is that $UGAP$ = Undirected Graph Accessibility Problem is in L and it thus also complete for L .

5 TC^0

Definition 6 (TC^0)

$$TC^0 := AC^0[MAJORITY] \quad (25)$$

where *MAJORITY* is the following problem:

$$MAJORITY(x_1, x_2, \dots, x_n) = \begin{cases} 1 & , \text{if } \sum_{i=1}^n x_i \geq \frac{n}{2} \\ 0 & , \text{otherwise} \end{cases} \quad (26)$$

Thesis (Maas, ~ 1980): Neuronal computation from one instant to the next is reasonably described by a TC^0 -computation.

Here a 'neuronal computation' means what a single neuron does. The thesis thus claims that a TC^0 -computation gives a model of the brain on a local level.

Proposition 2

$$AC^0 \subseteq TC^0 \subseteq NC^1 \subseteq L \quad (27)$$

Remark: If we accept the above thesis and the hierarchy in the proposition does not collapse ($TC^0 \neq NC^1$), then there are computational tasks (e.g. complete problems for NC^1) that can't be solved instantly by the brain. Examples of such tasks are $PUGAP = \text{Planar } UGAP$ and $PPAP = \text{Planar } PAP$, that are both complete for NC^1 .

Start of proof: It is enough to prove that $MAJORITY \in NC^1$ since by definition this problem is complete for TC^0 . But this will have to wait for the next lecture...

Exercise: $ADDITION \in AC^0$. Here our school-methods for addition do not work as the carries requires a sequential scan of the given bitstrings.

Remark: The following problems are complete for TC^0 : *MULTIPLICATION*, *INTEGER DIVISION* and *INTEGER SORTING*.