14.5.9

A language $L \subseteq \{0,1\}^*$ has *polynomial circuits* if there is a family $\mathcal{C} = (C_0, C_1, \ldots)$ such that (i) the size of C_n is at most $p(n)$ for some fixed polynomial p and (ii) $x = x_1x_2\ldots x_n \in L$ iff the output of C_n is true when the ith input variable is true if $x_i = 1$ and false otherwise.

A language A is *sparse* if its density function $\text{dens}_A(n) = |\{x \in A : |x| \leq n\}|$ is bounded by a fixed polynomial.

We shall show that a language L has polynomial circuits iff $L \in \mathbf{P}^A$ for some sparse language A.

The "if" direction. Assume that $L \in \mathbf{P}^A$ for some sparse language A such that A's density function is bounded by a polynomial $Q(n)$. Without loss of generality, assume that $A \subseteq \{0,1\}^*$. Let M be a deterministic machine that decides L in polynomial time using the oracle for A. Let the polynomial time bound function be $P(n)$. Now the maximum length of the oracle tape in any computation on input x is at most $P(|x|)$. Because A's density function is bounded by $Q(n)$, then the maximum number of strings in A of length $P(|x|)$ is at most $Q(P(|x|))$, a polynomial.

For each n there is a polynomial circuit $C_{|x|}$ such that $x = x_1x_2\ldots x_n \in L$ iff the output of the circuit is true for x. The circuit is like in the Cook's theorem (Theorem 8.2) except that we augment the circuit with "look-up" sub-circuits with $P(|x|)$ inputs deciding whether the contents on the oracle tape belong to A (the "look-up" sub-circuit is connected to the variables describing the current state of the oracle tape). Clearly the size of the "look-up" sub-circuit is $\mathcal{O}(P(|x|) \cdot Q(P(|x|)))$ (a disjunction of all codings of strings of length $P(|x|)$ belonging to A). Therefore the resulting circuit is polynomial and L has polynomial circuits if $L \in \mathbf{P}^A$ for some sparse language A.

The "only if" direction. Let L have polynomial circuits $\mathcal{C} = (C_0, C_1, \ldots)$. Let $\zeta(C_i)$ be the coding of circuit C_i in an alphabet Σ. Clearly $|\zeta(C_i)|$ is of polynomial length w.r.t. the size of C_i and thus w.r.t. i for any reasonable coding.

Consider the language A over Σ such that $x;y \in A$ iff $x \in \{0\}^*$ and y is a prefix of $\zeta(C_{|x|})$. Clearly A is sparse because for each n, there are no strings of length at most n in A concerning circuits C_m, $m > n$, and furthermore each coding of a circuit C_i, $i \leq n$, has at most polynomial number of prefixes w.r.t. n.

Now there is the following $\mathbf{P}^A$-machine M deciding L. On input x, M first counts the length n of x. It then systematically tests strings $0^n;\sigma$ where $\sigma \in \Sigma$ by using the oracle for a A until it finds a σ' which is a prefix of $\zeta(C_n)$. It then proceeds to test strings of form $0^n;\sigma'\sigma$ and so on until it has recovered the coding of C_n. This is done in time proportional to $|\Sigma| \cdot |\zeta(C_n)|$ which is a polynomial w.r.t. n. It then evaluates C_n on x and says yes iff the output gate was true.

Therefore $L \in \mathbf{P}^A$ for some sparse language A if it has polynomial circuits.

15.5.6 (a)

Let $\mathcal{C} = (C_0, C_1, \ldots)$ be a uniform family of circuits and let $f(n)$ and $g(n)$ be functions from integers to integers. The *parallel time* of $\mathcal{C}$ is at most $f(n)$ if for all n the depth of C_n is at most $f(n)$. We say that the *total work* of $\mathcal{C}$ is at most $g(n)$ if for all n the size of C_n is at most $g(n)$.

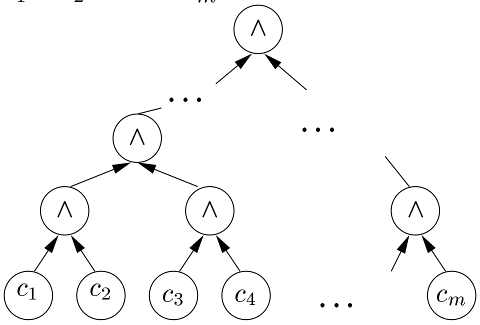
$\mathbf{NC}_j$ is the class of all languages $L \subseteq \{0,1\}^*$ such that there is uniform family of circuits $\mathcal{C}$ deciding L with $\mathcal{O}(\log^j n)$ parallel time and $\mathcal{O}(n^k)$ work.

The class $\mathbf{AC}_j$ is similar to $\mathbf{NC}$ except that the AND- and OR-gates in circuits can have unbounded fan-in.

We are to show that $\mathbf{NC}_i \subseteq \mathbf{AC}_i \subseteq \mathbf{NC}_{i+1}$.

The first inclusion holds trivially because each $\mathbf{NC}_i$ circuit is also an $\mathbf{AC}_i$ circuit.

Now take any AND- or OR-gate with fan-in $m > 2$. Replace this gate $c_1 \wedge c_2 \wedge \cdots \wedge c_m$ with:



Clearly the height of the replacing circuit is $\mathcal{O}(\log m)$ and the number of gates in it is $\mathcal{O}(m)$. Therefore, given an $\mathbf{AC}_i$ circuit C_n, we replace each gate with fan-in more than 2 with the equivalent set of gates with fan-in 2. The height of the resulting circuit is $\mathcal{O}(k \cdot \log n \cdot \log^i n)$ which equals to $\mathcal{O}(\log^{i+1} n)$ since the maximum fan-in (m) in C_n is at most the number n^k

of gates in C_n. On the other hand, the size of resulting circuit is at most $\mathcal{O}(n^k \cdot \max\{m\})$ where $\max\{m\}$ is the maximum fan-in in C_n which, as mentioned, is at most n^k — therefore the size of the resulting circuit is at most $\mathcal{O}(n^{2k})$, a polynomial. Thus the resulting circuit is a $\mathbf{NC}_{i+1}$ circuit. Since $\mathbf{AC}_i$ is uniform, there is a log-space algorithm generating the circuits in it. This same algorithm augmented with replacing the gates with fan-in more than 2 is clearly a log-space algorithm, too. Therefore, for each uniform family of $\mathbf{AC}_i$ circuits there is also a uniform family of $\mathbf{NC}_{i+1}$ circuits and the second inclusion holds.

From the inclusions $\mathbf{NC}_i \subseteq \mathbf{AC}_i \subseteq \mathbf{NC}_{i+1}$ is it easy to see that $\mathbf{NC} = \bigcup_i \mathbf{NC}_i = \bigcup_i \mathbf{AC}_i = \mathbf{AC}$.

17.3.3

We would like to show that $\mathbf{DP} \subseteq \mathbf{PP}$.

We know that $\mathbf{NP} \subseteq \mathbf{PP}$ (page 257 in the book). Furthermore, since $\mathbf{PP}$ is closed under complement, $\mathbf{coNP} \subseteq \mathbf{PP}$.

Take any language $L \in \mathbf{DP}$. Thus there exist languages $L_1 \in \mathbf{NP}$ and $L_2 \in \mathbf{coNP}$ such that $L = L_1 \cap L_2$. Take M_1 be the precise non-deterministic Turing machine deciding L_1 by using the $\mathbf{NP}$-acceptance condition. By the inclusions above, there exists a precise non-deterministic polynomial time Turing machine M_2 deciding L_2 by using the the $\mathbf{PP}$-acceptance condition. Let $p(\cdot)$ be the polynomial time bound of M_2. Now let M be the precise non-deterministic Turing machine that first executes M_1 on x. After each branch on which M_1 answers "no", M executes dummy computations of length $p(|x|)$ such that precisely $2^{p(|x|)-1}$ of them accept. On the other hand, after each branch on which M_1 answers "yes", M executes M_2 on x. Now we have three cases.

1. if $x \notin L_1$, then M never executes M_2 and therefore exactly half of the computations of M accept.
2. if $x \in L_1$ and $x \notin L_2$, at most half of the computations of M accept since after each accepting computation of M_1, at most half of the following computations of M_2 accept.
3. if $x \in L_1$ and $x \in L_2$, then after each accepting computation of M_1, more than half of the following computations of M_2 accept and since there is at least one accepting computation of M_1, more than half of the computations of M accept.

Thus M is a precise non-deterministic Turing machine that **PP**-accepts L and $\mathbf{DP} \subseteq \mathbf{PP}$.

17.3.6

Show that the following language is $\Delta_2\mathbf{P}$-complete: Given an instance of the TSP, is the optimum tour length odd? (Return false if no tour exists.)

We reduce from the following well-known $\Delta_2\mathbf{P}$-complete problem (Krentel, Journal of Computer and System Sciences, 36:490–509, 1988):

> Given a (3SAT) Boolean formula $\varphi(x_1, \ldots, x_n)$, is $x_n = 1$ in φ's lexicographically greatest satisfying assignment? (Return false if φ is not satisfiable.)

Use the construction of Theorem 17.5 in the book except that the "emergency exists" are removed. Now all the weights in the graph are zero except the following. The weights of arcs corresponding to literals $\neg x_i$ are 2^{2n+1-i} and the arcs corresponding to literals x_i have weight 2^i except that for x_n it is $2^n + 1$ (the only odd weight).

Now if φ cannot be satisfied, then the graph does not have a tour. If φ can be satisfied, then the optimum tour length must correspond to the lexicographically greatest satisfying assignment because of the weight distribution of arcs corresponding to the literals. This is because of the following. Take a tour corresponding to a satisfying truth assignment τ. Take any other satisfying truth assignment τ' that is lexicographically greater than τ. Let x_i be the first variable that is true in τ' but false in τ (there must exist one because τ' is greater than τ). Now the costs of the tours corresponding to τ and τ' are the same when only the weights corresponding to variables $x_1, \ldots, x_{i-1}$ are calculated. Since x_i is true in τ', the tour corresponding to τ' has an advantage of $2^{2n+1-i} - 2^i$ w.r.t. the tour corresponding to τ when only x_i is considered. Take the worst case when all the variables $x_{i+1}, \ldots, x_n$ are false in τ' and true in τ. Then the maximum advantage that τ can have on variables $x_{i+1}, \ldots, x_n$ w.r.t. τ' is $\sum_{j=i+1}^{n}(2^{2n+1-j} - 2^j) = \sum_{j=i+1}^{n} 2^{2n+1-j} - \sum_{j=i+1}^{n} 2^j < 2^{2n+1-i} - 2^i$. Therefore the cost of a tour corresponding to a lexicographically greater satisfying truth assignment is lesser. And because the weight of x_n is the only odd weight, the optimum tour is odd iff the last bit in the lexicographically greatest satisfying truth assignment for φ is 1.