Circuits

- A circuit has input bits $x_1, x_2, \ldots, x_n$ of some string $x = x_1 \ldots x_n$.
- It uses AND, OR and NOT gates.
- Each AND, OR gate has at most two inputs
- Notation: $C(x) = C(x_1, x_2, \ldots, x_n)$
- Size(C) = number of AND and OR gates
- Some times allow inputs $\neg x_1, \neg x_2, \ldots$.

- A Language L is decided by a family of circuits $C_0, C_1, \ldots$ (where C_n takes n input bits $x_1, \ldots, x_n$) iff for all n, for all x with $|x| = n$, $x \in L$ iff $C_n(x) = 1$
- $L \in Size(S(n))$, if it is decided by a family $C_0, C_1, \ldots$ where C_n has size $\leq S(n)$.

Theorem: Every language $L \in Size(O(2^n))$
proof: $Size(C_0) \leq 1$.
For C_{n+1}, note that $f(x_1, x_2, \ldots, x_{n+1}) =$
$(x_1$ AND $f(1, x_2, \ldots, x_{n+1}))$ OR
$((\neg x_1)$ AND $f(0, x_2, \ldots, x_{n+1}))$.

Thus, $Size(C_{n+1}) \leq 3 + 2 * Size(C_n)$
Thus, $Size(C_n) \leq 2 * 2^{n+1} - 3$, satisfies the constraints.

P/poly

For L to be in $\mathbf{P}/poly$, means it is accepted in Poly time by some machine M with advice/help of length at most poly-size in the length of the input. This help is same for all strings of the same length, but can be different for strings of different length. Formally:

- There is a two input TM M, which runs in poly-time
- For each length n, have advice string A_n (length bounded by polynomial in n).
- The polynomials are fixed for each L (independent of input x).
- x in L iff $M(x, A_{|x|})$ accepts.

Theorem: L in **P**/poly iff L can be decided by a polynomial size circuit
Proof: If L can be decided by polynomial size circuit, then using the coding of the circuit as advice to TM, one can decide L.

Now suppose L in **P**/poly (via M and A_n). Construct a circuit as follows:
On input x, we additionally have "fixed" bits of advice $A_{|x|}$.
Then, we can consider calculating $T_0(x, A_{|x|}), T_1(x, A_x), \ldots$ where $T_i(x, A_{|x|})$ is the configuration of the TM after i steps of computation.

Initial configuration $T_0(x, A_{|x|})$ is easy to describe.

Using poly-size circuit one can calculate $T_{i+1}(x, A_{|x|})$ from $T_i(x, A_{|x|})$.

From $T_{p(n)}(X, A_{|x|})$ using poly-size circuit one can calculate the answer.

So summing up the above circuit sizes, in total a polynomial size circuit can decide L.

Note that $\mathbf{P}/poly$ is not contained in $\mathbf{P}$ or even
$DTIME(2^{2^{2^{2^n}}})$
Think of a language which depends only on the input size and $\ldots$!

Is $NP \subseteq \mathbf{P}/poly$? Unlikely.
Theorem: If $NP \subseteq \mathbf{P}/poly$, then $PH \subseteq \Sigma_2^p$.
We will show that if $NP \subseteq \mathbf{P}/poly$, then $\Pi_2^p \subseteq \Sigma_2^p$.
First suppose $NP \subseteq \mathbf{P}/poly$, then for the languages $L \in NP$, we can find the "y" in $(\exists y)[P(x, y)]$ as in the characterization of NP, using poly size circuits. For this, one can do as follows:
Suppose $x \in L$ iff $(\exists y : |y| = p(|x|))[P(x, y)]$.
Let $L' = \{(x, z\#^{p(|x|)-|z|}) :$ there exists $z',\ P(x, zz')\}$.
Then, $L' \in NP$, and thus in $\mathbf{P}/poly$.
Let the above be witnessed by machine M.
Now, using above M one can construct M', in $\mathbf{P}/poly$, which on input x can iteratively find y_1, y_1y_2, $y_1y_2y_3 \ldots$, $y_1y_2 \ldots y_{p|x|}$, if such a y exists.

Now let $L \in \Pi_2^p$. Then
$x \in L$ iff $(\forall y)(\exists z)[Q(x, y, z)]$, where y, z are poly-size bounded with respect to $|x|$, and Q is a poly-time computable predicate.
Let $L' = \{(x, y) : (\exists z)Q(x, y, z)\}$.
Now, L' is in NP. Thus, there is a poly-size circuit which computes z from (x, y).
Now, $x \in L$ iff $\exists C$ of size polynomial in length of x, such that for all y, $Q(x, y, C(x, y))$, where length of y and C are polynomially bounded.
Thus, $L \in \Sigma_2^p$.

BPP

Theorem: $BPP \subseteq P/poly$
Proof: If L in BPP, then there is an algorithm $A(x, r)$ ($|r|$ polynomial in $|x|$) which outputs correct value for $L(x)$ with probability at least $1 - \frac{1}{2^{|x|+2}}$.
Thus, for any particular length n, for at least $3/4$-th of the corresponding possible r of length polynomial in n, for all x, $A(x, r)$ gives the correct answer.
Thus, we can use such r as the help bits!