P and Polynomial-Time

Informal Definition of $\mathcal{P}$

The collection of all problems that can be solved in polynomial time is called $\mathcal{P}$.

Example. The language of all binary palindromes is decidable in polynomial time.

Run-Time of a TM

We analyze the *worst-case*: what matters is whether *every* instance can be done quickly. A TM ***runs in time*** $T(n)$ if, for all inputs w, it halts within $T(|w|)$ steps.

n always denotes the input length.

Order Notation

Recall that ***big-O*** notation says how the worst-case running time grows as n gets large.

For example, we say a TM runs in $O(n^2)$ time, or "***order*** n^2" time, if there is c such that the TM runs in at most cn^2 steps for *any* input of length n.

More Formal Definition of $\mathcal{P}$

A language L is in $\mathcal{P}$ if there is k and TM that decides L that runs in time $O(n^k)$.

Example Again. The language of all binary palindromes is in $\mathcal{P}$. The TM we built earlier—that compares first and last character and repeats—runs in time $O(n^2)$.

The Definition of $\mathcal{P}$ is Robust

It can be argued that Church's thesis carries over to polynomial-time. That is, the definition of $\mathcal{P}$ is the same whether we think of TMs or of normal computer algorithms.

Example: Boolean Formula

A ***boolean formula*** consists of variables and negated variables (collectively ***literals***), and the operations "and" and "or". We use $\vee$ for "or", $\wedge$ for "and", and $\bar{x}$ for "not x". E.g.

$$x \wedge (x \vee y \vee \bar{z}) \wedge (\bar{x} \vee \bar{y})$$

An ***assignment*** is setting each variable to either TRUE or FALSE. For example, if x and y are TRUE and z is FALSE, the above formula is FALSE.

True Boolean Formulas

$\texttt{TRUEBF} = \{ \langle \phi, \psi \rangle : \phi$ is boolean formula made
TRUE by assignment ψ $\}$

A boolean formula can be parsed and evaluated like a normal arithmetic expression using a single stack. Thus TRUEBF is in $\mathcal{P}$.

Practice: Acceptance Problem for NFAs

Show that $A_{nfa} = \{ \langle M, w \rangle : M$ is NFA accepting $w \}$ is in $\mathcal{P}$.

Solution to Practice

One cannot convert the NFA to a DFA, since the DFA can be exponentially large. Instead, use the idea that motivated the subset construction.

Simulate the NFA by keeping track of at each step the list of states the NFA could be in. Updating this list for a single symbol takes polynomial time.