More on reductions; nondeterministic computation

Polynomial time reductions enable us to compare tractability of languages.

THEOREM Suppose $L_1 \subseteq \Sigma_1^*$ and $L_2 \subseteq \Sigma_2^*$ are languages. If L_1 is polynomial-time reducible to L_2, and L_2 is in the class P, then L_1 is also in the class P.

Notes:

1. If L_2 recognized in time n^d and f computable in time n^e then L_1 recognized in time $n^{de} + 2n^e$; this might well be the best we can do.

2. P is closed under reductions.

3. If $L_1 \leq_{\mathsf{P}} L_2$ and $L_1 \notin P$ then $L_2 \notin P$.

Naïve search and P:

EULERIAN CYCLES

INSTANCE: A finite undirected graph G.

QUESTION: Does G have an Eulerian cycle, i.e., can we start at a vertex v visit every edge exactly once and return to v?

Algorithm: Try all possible permutations of the edges and for each permutation test it to see if it does the job.

Note: Testing phase is easy.

Real Algorithm (Euler): Answer is 'yes' if and only if graph is connected and each vertex has an even number of edges attached.

Further Examples:

PERFECT MATCHINGS

INSTANCE: Bipartite graph B with equal numbers of vertices in each partition.

QUESTION: Does B have a perfect matching?

Efficient Algorithm: Based on Flow Networks.

Contrast: SAT and CLIQUE.

- Proposed solutions easy to check.
- Nobody knows how to get around naïve search.

Nondeterministic Turing machines: Same as ordinary TMs but now each state-symbol pair can have more than one instruction.

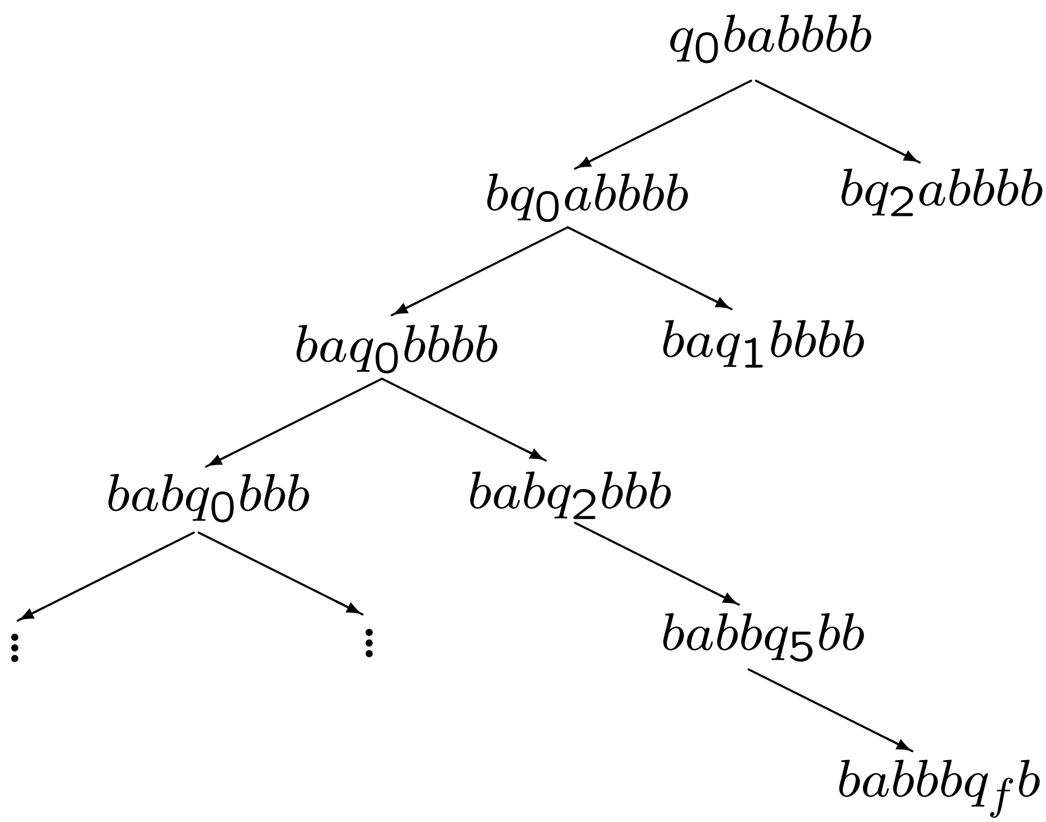
- δ is a relation rather than a partial function.

- At each step, machine chooses nondeterministically a next step (if any).

- Machine accepts input x if and only if there is at least one computation leading to final state.

Example: $L \subset \{a, b\}^*$ all strings that contain aaa or bbb as a substring.

$$
\begin{array}{lll}
(q_0, a, q_0, a, R) & (q_0, b, q_0, b, R) & (q_0, a, q_1, a, R) \\
(q_0, b, q_2, b, R) & (q_1, a, q_3, a, R) & (q_3, a, q_f, a, R) \\
(q_2, b, q_5, b, R) & (q_5, b, q_f, b, R) & (q_3, b, q_0, b, L)
\end{array}
$$



Simulating NTMs by DTMs:

THEOREM If a language L is accepted by a nondeterministic Turing machine M, then L is accepted by a deterministic Turing machine $\widehat{M}$.

input:	x
choice sequence:	
simulation of M:	

Choice sequence:

- Number all nondeterministic choices up to a given depth.
- Choose moves specified in the sequence.

The language class NP:

M a NTM, input alphabet Σ.

M is *of time complexity* $T(n)$ means:

• M makes at most $T(n)$ transitions before halting, for all inputs $x \in \Sigma^n$.

M is *polynomial time* means

• it is of time complexity $p(n)$ for some polynomial p.

NP is class of all languages accepted by polynomial-time NTMs

Example languages in NP:

1. SAT, CLIQUE.

2. EULERIAN CYCLES, PERFECT MATCHINGS.

3. Every language from P.

Open problem: Have

$$\mathsf{P} \subseteq \mathsf{NP}.$$

Is this a strict containment?

Look ahead: We will see that if $L \in \mathsf{NP}$ then $L \leq_{\mathsf{P}}$ SAT.

• Thus CLIQUE $\leq_{\mathsf{P}}$ SAT so CLIQUE $\in \mathsf{P}$ if and only if SAT $\in \mathsf{P}$.

• similar situation holds for many hundreds of natural problems.