

# Defining P and NP

- Decision Problems : We will start out with decision problems. As mentioned in Lecture 24, such problems have only two answers: yes or no.

Formally, a decision problem  $\Pi$  consists of a set  $D_\Pi$  of all instances and a subset  $Y_\Pi \subseteq D_\Pi$  of yes instances.

- Solving a decision problem is similar to solving the word acceptance problem in the theory of languages.

Using the notation of Garey and Johnson, let  $\Sigma$  be some alphabet and  $L \subseteq \Sigma^*$  a language. Let's now consider a Turing Machine for recognizing  $L$ . (From now on, we will refer to the Turing Machines of the sort introduced in Lecture 12 as Deterministic Turing Machines. The acronym DTM will represent such machines. One step of computation by a DTM amounts to the scan-head looking at a symbol  $s_j$  in state  $q_k$ , and then possibly erasing  $s_j$  and replacing it with  $s_k$ , moving one square to the left or right and transitioning into state  $q_\ell$ . So our DTM is a quintuple Turing Machine.)

- A DTM will be considered to solve the language recognition problem for a particular language  $L \subseteq \Sigma^*$  if it halts in state  $q_y$  when  $x \in L$  and in state  $q_N$  when  $x \in \Sigma^* - L$  where  $x$  is the word placed on the tape.

- We will now formalize the 1-1 relationship between decision problems and languages. Let there exist an encoding scheme  $e$  to express an instance  $I \in D_\Pi$  as a word  $x$  on some alphabet  $\Sigma$ . We can then define a language  $L$  that would correspond to the subset  $Y_\Pi$  by

$$L(\Pi, e) = \{x \in \Sigma^* \mid x \text{ is the encoding under } e \text{ of an instance } I \in Y_\Pi\}$$

- Let's now say that a DTM  $M$ , with its special states  $q_y$  and  $q_N$ , can be used to recognize the language  $L(\Pi, e)$ . We can now formally define the Time Complexity Function that was introduced in Lecture 24:

$$T_M(n) = \max \left\{ m \mid \begin{array}{l} \text{there exists an } x \in \Sigma^* \text{ with } |x|=n \text{ such that} \\ M \text{ takes } m \text{ steps to halt in } q_y \text{ or } q_N \end{array} \right\}$$

Obviously,  $T_M : \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$  where  $\mathbb{Z}^+$  is the set of all positive integers.

- As you'd expect, the DTM  $M$  must halt in  $q_y$  for all  $x \in L(\Pi, e)$  and in state  $q_N$  for all  $x \in \Sigma^* - L(\Pi, e)$ . Henceforth, we will refer to the set of quintuples that constitute the DTM as the DTM program  $M$ .

- We will refer to  $M$  as a polynomial time DTM program if there exists a polynomial  $p$  such that for all  $n \in \mathbb{Z}^+$  we have

$$T_M(n) \leq p(n)$$

- We are now ready to first define the class P of languages and to then define the class P of decision problems. (A class is the same thing as a set.)
- Here is the definition of class P of languages :

$$P = \{ L \mid \text{there exists a polynomial time DTM program } M \text{ such that } M \text{ halts in } q_y \text{ for all } x \in L \text{ and in state } q_N \text{ for all } x \in \Sigma^* - L \}$$

- We say a decision problem  $\Pi$  belongs to class P under the encoding scheme  $e$  if

$$L(\Pi, e) \in P$$

## Nondeterministic Algorithms

- Earlier in Lecture 24 we introduced the notion of a nondeterministic computer. A nondeterministic computer carries out an unbounded number of simultaneous calculations, with each calculation thread consisting of randomly guessing a solution to the problem and then verifying that guess. The computing that is carried out in each thread — constructing a guess and then verifying the guess — is said to be the work of a nondeterministic algorithm.
- Therefore, a nondeterministic algorithm is composed of two separate stages :
  - The Guessing Stage, and
  - The Checking Stage.
- Given a problem instance  $I \in D_\Pi$ , the Guessing stage merely outputs some structure  $S$  as a guess.
- Using  $I$  and  $S$  as input, the Checking stage ~~computes~~ executes the steps
  - either eventually halts with answer 'yes'.
  - or, eventually halts with answer 'no'.
  - or, computes for ever without halting.

By contrast, a deterministic algorithm halts for all inputs, for  $I \in Y_\Pi$ , as well as for  $I \in D_\Pi - Y_\Pi$ .

## What Does It Mean When We Say That a Nondeterministic Algorithm Solves a Decision Problem :

We say a nondeterministic algorithm solves a decision problem  $\Pi$  if the following two properties hold for all instances  $I \in D_\Pi$  :

- If  $I \in Y_\Pi$ , then there exists some guess  $S$  that will result in the Checking stage to respond 'yes' when supplied with  $I$  and  $S$ .
- If  $I \notin Y_\Pi$ , there exists NO guess  $S$  that will result in the Checking stage to respond with 'yes' when supplied with  $I$  and  $S$ .

- We say that a nondeterministic algorithm solves a decision problem  $\Pi$  in polynomial time if there exists a polynomial  $p$  such that, for every instance  $I \in Y_\Pi$ , there is some guess  $S$  that leads the checking stage to respond 'yes' for the supplied  $I$  and  $S$  within time  $p(\text{length}(I))$ .
- Definition of class NP : The class NP is defined to be the class of all decision problems  $\Pi$  that, under reasonable encoding schemes, can be solved by polynomial time nondeterministic algorithms.