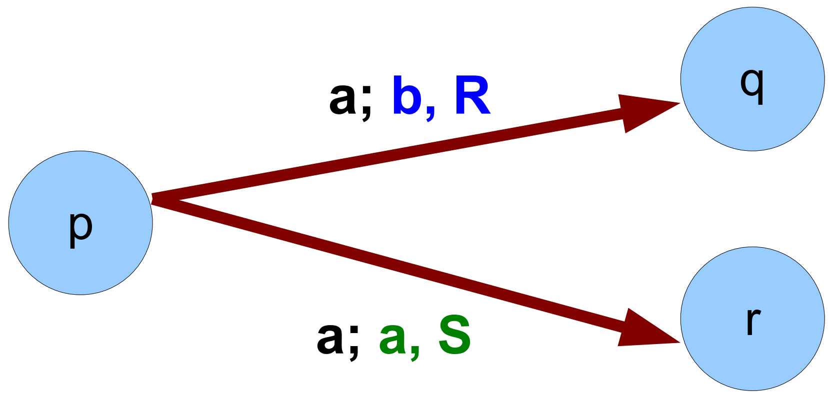
Non-deterministic Turing Machines

- Recall: a Turing machine is non-deterministic if its transitions may allow choices



- Every deterministic TM (DTM) is a trivial case of a non-deterministic TM (NDTM)
- Every NDTM can be converted to a DTM but the resulting DTM may be exponentially larger

Decidability vs. Computational Complexity

- Decidability is focused only on whether a problem (or equivalently, a language) can be solved or decided by an algorithm (or TM) in a finite number of steps
- Computational complexity studies the amount of resources (time, space, processors, and their possible trade-offs) needed to solve a problem (or equivalently, decide a language)

Some examples

- Given a string x of length n over {0,1} representing a binary integer, we can build a DTM that computes x+1 in about 2n steps; we say the algorithm runs in O(n) time
- Adding 2 binary integers can also be done in O(n) time, if we use 3 tapes (2 tapes for the inputs and another tape for the sum)

Running time of algorithms

- The same basic algorithm can have different running times when using different models of computation
- Example: The bubblesort algorithm obviously runs slower on TuringKara than on a PC using a real programming language
- However, they are both polynomial-time algorithms $O(n^k)$ when sorting n numbers

Polynomial-time solvable problems

- A problem is polynomial-time solvable if an instance x of length n can be solved by a TM in $O(n^k)$ steps for some integer k
- We define

  P = the class of all problems that are solvable in polynomial-time by a DTM

  NP = the class of all problems that are solvable in polynomial-time by a NDTM

NDTMs and DTMs

- Trivially, a NDTM that runs in polynomial-time can be converted to a DTM that runs in exponential-time using backtracking (trying all possible branches)
- Equivalently, one can convert a polynomial-time NDTM to a polynomial-time DTM by using an exponential number of processors to try all possible branches
- But for large problem instances, exponential-time and exponential-number-of-processors are both unreasonable models in practice

NDTM example $\quad z = \text{babaaaabba}$

- Consider the language over $\Sigma = \{ a, b \}$:

  $L = \{ xy : x, y \text{ in } \Sigma^+ \text{ and } x \text{ is any permutation of } y \}$

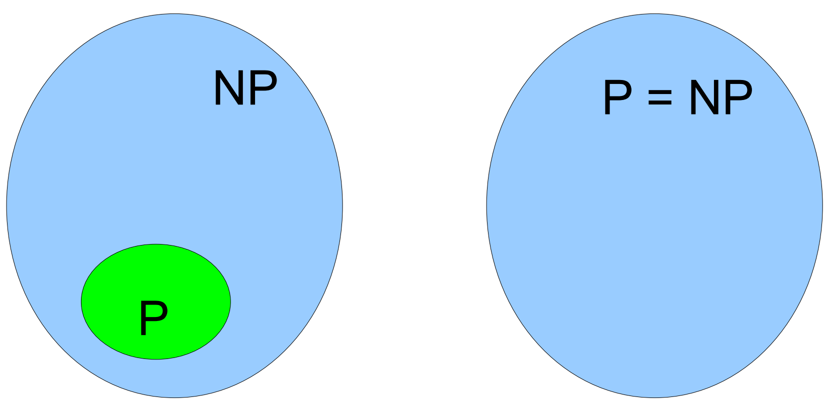
  $= \{ aa, bb, aaaa, abab, abba, baab, baba, \ldots \}$

- One can construct a DTM that decides if an arbitrary input string z is a member in L by identifying the two halves of the string (x and y, where z = xy), sort both halves and check if sorted(x) = sorted(y)
- An NDTM is easier: guess where the midpoint of z is, guess how x and y are shuffled separately and shuffle them, then verify in linear time that shuffled(x) = shuffled(y)

A famous open problem*

(posed by Stephen Cook in 1971)

- P $\subseteq$ NP, but is P $\subset$ NP? Or is P $=$ NP?
- If P = NP, then any problem that can be solved by a NDTM in polynomial time can also be solved by a DTM in polynomial time
- Most researchers believe P $\subset$ NP, but no one has a found a proof yet



* a $1M prize awaits the prover of either result, see claymath.org/millenium/

NP-complete problems

- NP-complete problems are the hardest problems in the class NP
- An NP-complete problem F satisfies the conditions:
  - F is a member in the class NP
  - Every problem in NP is reducible to F in polynomial-time
  - (Or equivalently by transitivity, some other known NP-complete problem reduces to F in polynomial-time)
  - IF we can solve an NP-complete problem in polynomial-time, we (indirectly) solve all problems in NP in polynomial time. (But still a big IF.)

Some NP-complete problems

- SATISFIABILITY: Given a Boolean expression in conjunctive normal form involving n variables, and the operators (not, and, or), is there a truth assignment to the variables that makes the expression true?

  Easy to solve by trying all possible T/F combinations, but is there a solution significantly better than $O(2^n)$?

- TRAVELING SALESMAN: Given n cities, distances between them, and a bound B, is there a tour that visits all the cities exactly once and with total distance $< B$?

  Easy to solve by trying all $(n-1)!$ permutations but is there a solution that runs in polynomial-time?