Time Complexity

- The multistring TM is the basis of our notion of the time expended by TM computations.
- If for a k-string TM M and input x, the TM halts after t steps, then the **time required by M on input x** is t.
- If $M(x) = \nearrow$, then the time required by M on x is ∞.
- Machine M **operates within time** $f(n)$ for $f : \mathbb{N} \to \mathbb{N}$ if for any input string x, the time required by M on x is at most $f(|\,x\,|)$.
  - $|\,x\,|$ is the length of string x.
  - Function $f(n)$ is a **time bound** for M.

Time Complexity Classes[a]

- Suppose language $L \subseteq (\Sigma - \{\sqcup\})^*$ is decided by a multistring TM operating in time $f(n)$.
- We say $L \in \text{TIME}(f(n))$.
- $\text{TIME}(f(n))$ is the set of languages decided by TMs with multiple strings operating within time bound $f(n)$.
- $\text{TIME}(f(n))$ is a **complexity class**.
  - PALINDROME is in $\text{TIME}(f(n))$, where $f(n) = O(n)$.

[a]Hartmanis and Stearns (1965), Hartmanis, Lewis, and Stearns (1965).

The Simulation Technique

Theorem 2 *Given any k-string M operating within time $f(n)$, there exists a (single-string) M' operating within time $O(f(n)^2)$ such that $M(x) = M'(x)$ for any input x.*

- The single string of M' implements the k strings of M.
- Represent configuration $(q, w_1, u_1, w_2, u_2, \ldots, w_k, u_k)$ of M by configuration

$$(q, \triangleright w_1' u_1 \triangleleft w_2' u_2 \triangleleft \cdots \triangleleft w_k' u_k \triangleleft \triangleleft)$$

  of M'.
  - $\triangleleft$ is a special delimiter.
  - w_i' is w_i with the first and last symbols "primed."

The Proof (continued)

- The initial configuration of M' is

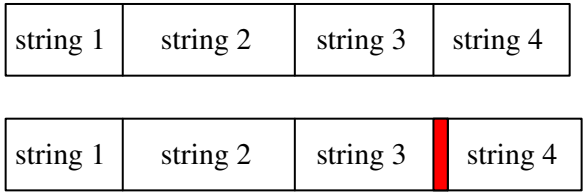
$$(s, \triangleright \triangleright' x \triangleleft \overbrace{\triangleright' \triangleleft \cdots \triangleright' \triangleleft}^{k-1 \text{ pairs}} \triangleleft).$$

- To simulate each move of M:
  - M' scans the string to pick up the k symbols under the cursors.
    - The states of M' must include $K \times \Sigma^k$ to remember them.
    - The transition functions of M' must also reflect it.
  - M' then changes the string to reflect the overwriting of symbols and cursor movements of M.

The Proof (continued)

- It is possible that some strings of M need to be lengthened.
  - The linear-time algorithm on p. 22 can be used for each such string.
- The simulation continues until M halts.
- M' erases all strings of M except the last one.
- Since M halts within time $f(|\,x\,|)$, none of its strings ever becomes longer than $f(|\,x\,|)$.[a]
- The length of the string of M' at any time is $O(kf(|\,x\,|))$.

[a]We tacitly assume $f(n) \geq n$.



The Proof (concluded)

- Simulating each step of M takes, *per string of* M, $O(kf(|\,x\,|))$ steps.
  - $O(f(|\,x\,|))$ steps to collect information.
  - $O(kf(|\,x\,|))$ steps to write and, if needed, to lengthen the string.
- M' takes $O(k^2 f(|\,x\,|))$ steps to simulate each step of M.
- As there are $f(|\,x\,|)$ steps of M to simulate, M' operates within time $O(k^2 f(|\,x\,|)^2)$.

Linear Speedup[a]

Theorem 3 *Let $L \in TIME(f(n))$. Then for any $\epsilon > 0$, $L \in TIME(f'(n))$, where $f'(n) = \epsilon f(n) + n + 2$.*

- If $f(n) = cn$ with $c > 1$, then c can be made arbitrarily close to 1.
- If $f(n)$ is superlinear, say $f(n) = 14n^2 + 31n$, then the constant in the leading term (14 in this example) can be made arbitrarily small.
  - *Arbitrary* linear speedup can be achieved.
  - This justifies the asymptotic big-O notation.

[a]Hartmanis and Stearns (1965).

P

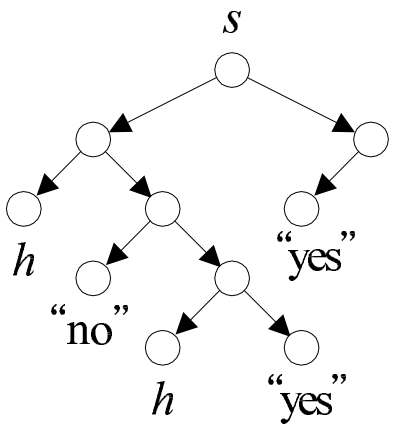
- By the linear speedup theorem, any polynomial time bound can be represented by its leading term n^k for some $k \geq 1$.
- If L is a polynomially decidable language, it is in $\text{TIME}(n^k)$ for some $k \in \mathbb{N}$.
  - Clearly, $\text{TIME}(n^k) \subseteq \text{TIME}(n^{k+1})$.
- The union of all polynomially decidable languages is denoted by P:
$$\text{P} = \bigcup_{k>0} \text{TIME}(n^k).$$
- Problems in P can be efficiently solved.

Nondeterminism[a]

- A **nondeterministic Turing machine** (**NTM**) is a quadruple $N = (K, \Sigma, \Delta, s)$.
- K, Σ, s are as before.
- $\Delta \subseteq K \times \Sigma \to (K \cup \{h, \text{“yes”}, \text{“no”}\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}$ is a relation, not a function.
  - For each state-symbol combination, there may be more than one next steps—or none at all.
- A configuration yields another configuration in one step if there *exists* a rule in Δ that makes this happen.

[a]Rabin and Scott (1959).

Computation Tree and Computation Path



Decidability under Nondeterminism

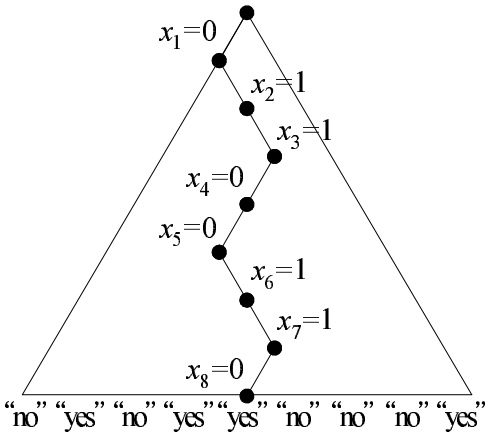
- Let L be a language and N be an NTM.
- N **decides** L if for any $x \in \Sigma^*$, $x \in L$ if and only if there is a sequence of valid configurations that ends in “yes.”
  - It is not required that the NTM halts in all computation paths.
  - If $x \notin L$, no nondeterministic choices should lead to a “yes” state.
- What is key is the algorithm's overall behavior not whether it gives a correct answer for each particular run.
- Determinism is a special case of nondeterminism.

A Nondeterministic Algorithm for Satisfiability

ϕ is a boolean formula with n variables.

1: **for** $i = 1, 2, \ldots, n$ **do**
2: Guess $x_i \in \{0, 1\}$; {Nondeterministic choice.}
3: **end for**
4: {Verification:}
5: **if** $\phi(x_1, x_2, \ldots, x_n) = 1$ **then**
6: "yes";
7: **else**
8: "no";
9: **end if**

The Computation Tree for Satisfiability



Analysis

- The algorithm decides language $\{\phi : \phi \text{ is satisfiable}\}$.
  - The computation tree is a complete binary tree of depth n.
  - Every computation path corresponds to a particular truth assignment out of 2^n.
  - ϕ is satisfiable if and only if there is a computation path (truth assignment) that results in "yes."
- General paradigm: Guess a "proof" and then verify it.

The Traveling Salesman Problem

- We are given n cities $1, 2, \ldots, n$ and integer distances d_{ij} between any two cities i and j.
- Assume $d_{ij} = d_{ji}$ for convenience.
- The **traveling salesman problem** (TSP) asks for the total distance of the shortest tour of the cities.
- The decision version TSP (D) asks if there is a tour with a total distance at most B, where B is an input.

A Nondeterministic Algorithm for TSP (D)

1: **for** $i = 1, 2, \ldots, n$ **do**
2: Guess $x_i \in \{1, 2, \ldots, n\}$; {The ith city.}
3: **end for**
4: $x_{n+1} := x_1$;
5: {Verification stage:}
6: **if** $x_1, x_2, \ldots, x_n$ are distinct and $\sum_{i=1}^{n} d_{x_i, x_{i+1}} \leq B$ **then**
7: "yes";
8: **else**
9: "no";
10: **end if**

Time Complexity under Nondeterminism

- Nondeterministic machine N decides L **in time** $f(n)$, where $f : \mathbb{N} \to \mathbb{N}$, if
  - N decides L, and
  - for any $x \in \Sigma^*$, N does not have a computation path longer than $f(|\, x \,|)$.
- We charge only the "depth" of the computation tree.

Time Complexity Classes under Nondeterminism

- NTIME($f(n)$) is the set of languages decided by NTMs within time $f(n)$.
- NTIME($f(n)$) is a complexity class.

NP

- Define
$$\text{NP} = \bigcup_{k>0} \text{NTIME}(n^k).$$
- Clearly P $\subseteq$ NP.
- Think of NP as efficiently *verifiable* problems.
  - Boolean satisfiability (SAT).
  - TSP (D).
- The most important open problem in computer science is whether P = NP.

Simulating Nondeterministic TMs

Theorem 4 *Suppose language L is decided by an NTM N in time $f(n)$. Then it is decided by a 3-string deterministic TM M in time $O(c^{f(n)})$, where $c > 1$ is some constant depending on N.*

- On input x, M goes down every computation path of N using *depth-first* search (but M does *not* know $f(n)$).
  - As M is time-bounded, the depth-first search will not run indefinitely.

The Proof (concluded)

- If some path leads to "yes," then M enters the "yes" state.
- If none of the paths leads to "yes," then M enters the "no" state.

Corollary 5 $\text{NTIME}(f(n))) \subseteq \bigcup_{c>1} \text{TIME}(c^{f(n)})$.

Undecidability

It seemed unworthy of a grown man
to spend his time on such trivialities,
but what was I to do?
— Bertrand Russell (1872–1970),
Autobiography, Vol. I

Universal Turing Machine[a]

- A **universal Turing machine** U interprets the input as the *description* of a TM M concatenated with the *description* of an input to that machine, x.
  - Both M and x are over the alphabet of U.
- U simulates M on x so that

$$U(M;x) = M(x).$$

- U is like a modern computer, which executes any valid machine code, or a Java Virtual machine, which executes any valid bytecode.

[a]Turing (1936).

The Halting Problem

- **Undecidable problems** are problems that have no algorithms or languages that are not recursive.
- We now define a concrete undecidable problem, the **halting problem**:

$$H = \{M;x : M(x) \neq \nearrow\}.$$

  - Does M halt on input x?

H Is Recursively Enumerable

- Use the universal TM U to simulate M on x.
- When M is about to halt, U enters a "yes" state.
- If $M(x)$ diverges, so does U.
- This TM accepts H.
- Membership of x in any recursively enumerative language accepted by M can be answered by asking

$$M;x \in H?$$

H Is Not Recursive

- Suppose there is a TM M_H that *decides* H.
- Consider the program $D(M)$ that calls M_H:

```
1: if M_H(M;M) = "yes" then
2:     ↗; {Writing an infinite loop is easy, right?}
3: else
4:     "yes";
5: end if
```

- Consider $D(D)$:
  - $D(D) = \nearrow \Rightarrow M_H(D;D) =$ "yes" $\Rightarrow D; D \in H \Rightarrow D(D) \neq \nearrow$, a contradiction.
  - $D(D) =$ "yes" $\Rightarrow M_H(D;D) =$ "no" $\Rightarrow D; D \notin H \Rightarrow D(D) = \nearrow$, a contradiction.

Comments

- Two levels of interpretations of M:
  - A sequence of 0s and 1s (data).
  - An encoding of instructions (programs).
- There are no paradoxes.
  - Concepts should be familiar to computer scientists.
  - Supply a C compiler to a C compiler, a Lisp interpreter to a Lisp interpreter, etc.

Self-Loop Paradoxes

Cantor's Paradox (1899): Let T be the set of all sets.

- Then $2^T \subseteq T$, but we know $|2^T| > |T|$ (Cantor's theorey)!

Eubulides: The Cretan says, "All Cretans are liars."

Liar's Paradox: "This sentence is false."

Sharon Stone in *The Specialist* (1994): "I'm not a woman you can trust."