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:

$$\text{P} = \bigcup_{k>0} \text{TIME}(n^k).$$

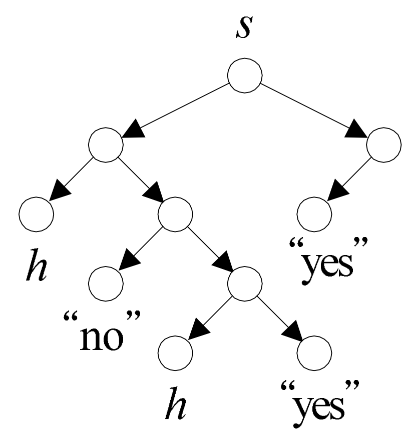
- Problems in P can be efficiently solved.

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

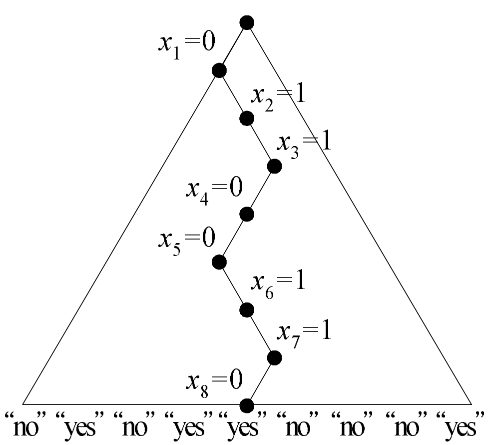
- Let L be a language and N be an NTM.
- N **decides** L if for any $x \in \Sigma^*$, $x \in L$ if and only if there is a sequence of valid configurations that ends in “yes.”
  - It is not required that the NTM halts in all computation paths.
  - If $x \notin L$, no nondeterministic choices should lead 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.

```
for i = 1, 2, ..., n do
    Guess x_i ∈ {0, 1}; {Nondeterministic choice.}
end for
{Verification:}
if φ(x_1, x_2, ..., x_n) = 1 then
    "yes";
else
    "no";
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)

for $i = 1, 2, \ldots, n$ **do**
  Guess $x_i \in \{1, 2, \ldots, n\}$; {The ith city.}
end for
$x_{n+1} := x_1$;
{Verification stage:}
if $x_1, x_2, \ldots, x_n$ are distinct and $\sum_{i=1}^{n} d_{x_i, x_{i+1}} \leq B$ **then**
  "yes";
else
  "no";
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 $\text{P} \subseteq \text{NP}$.
- Think of NP as efficiently *verifiable* problems.
  - Boolean satisfiability (SAT).
  - TSP (D).
- The most important open problem in computer science is whether $\text{P} = \text{NP}$.

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:

if $M_H(M; M) =$ "yes" **then**
    $\nearrow$; {Writing an infinite loop is easy, right?}
else
    "yes";
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."

More Undecidability

Theorem 6 *$H^* = \{M : M$ halts on all inputs$\}$ is undecidable*

- Given $M; x$, we construct the following machine:

$$M_x(y) : \text{if } y = x \text{ then } M(x) \text{ else halt.}$$

- M_x halts on all inputs if and only if M halts on x.
- In other words, $M; x \in H$ if and only if $M_x \in H^*$.
- So if the said language were recursive, H would be recursive, a contradiction.
- This technique is called **reduction**.

Reductions in Proving Undecidability

- Suppose we are asked to prove L is undecidable.
- Language H is known to be undecidable.
- We try to find a computable transformation (or reduction) R such that that[a]

$$\forall x (R(x) \in L \text{ if and only if } x \in H).$$

- We can answer "$x \in H$?" for any x by asking $R(x) \in L$?
- This suffices to prove that L is undecidable.

[a]Contributed by Mr. Tai-Dai Chou (J93922005) on May 19, 2005.

Complements of Recursive Languages

Lemma 7 *If L is recursive, then so is $\bar{L}$.*

- Let L be decided by M (which is deterministic).
- Swap the "yes" state and the "no" state of M.
- The new machine decides $\bar{L}$.

Recursive and Recursively Enumerable Languages

Lemma 8 *L is recursive if and only if both L and $\bar{L}$ are recursively enumerable.*

- Suppose both L and $\bar{L}$ are recursively enumerable, accepted by M and $\bar{M}$, respectively.
- Simulate M and $\bar{M}$ in an *interleaved* fashion.
- If M accepts, then $x \in L$ and M' halts on state "yes."
- If $\bar{M}$ accepts, then $x \notin L$ and M' halts on state "no."