BU CS 332 – Theory of Computation

https://forms.gle/dKhaa9iFatDd2d2ZA

Lecture 20:

- Complexity Class NP
- Nondeterminstic TMs vs. Deterministic Verifiers

Reading:

Sipser Ch 7.3

Final
Th 5/7 12-2PM

Alexander Poremba & Mark Bun

April 16, 2026

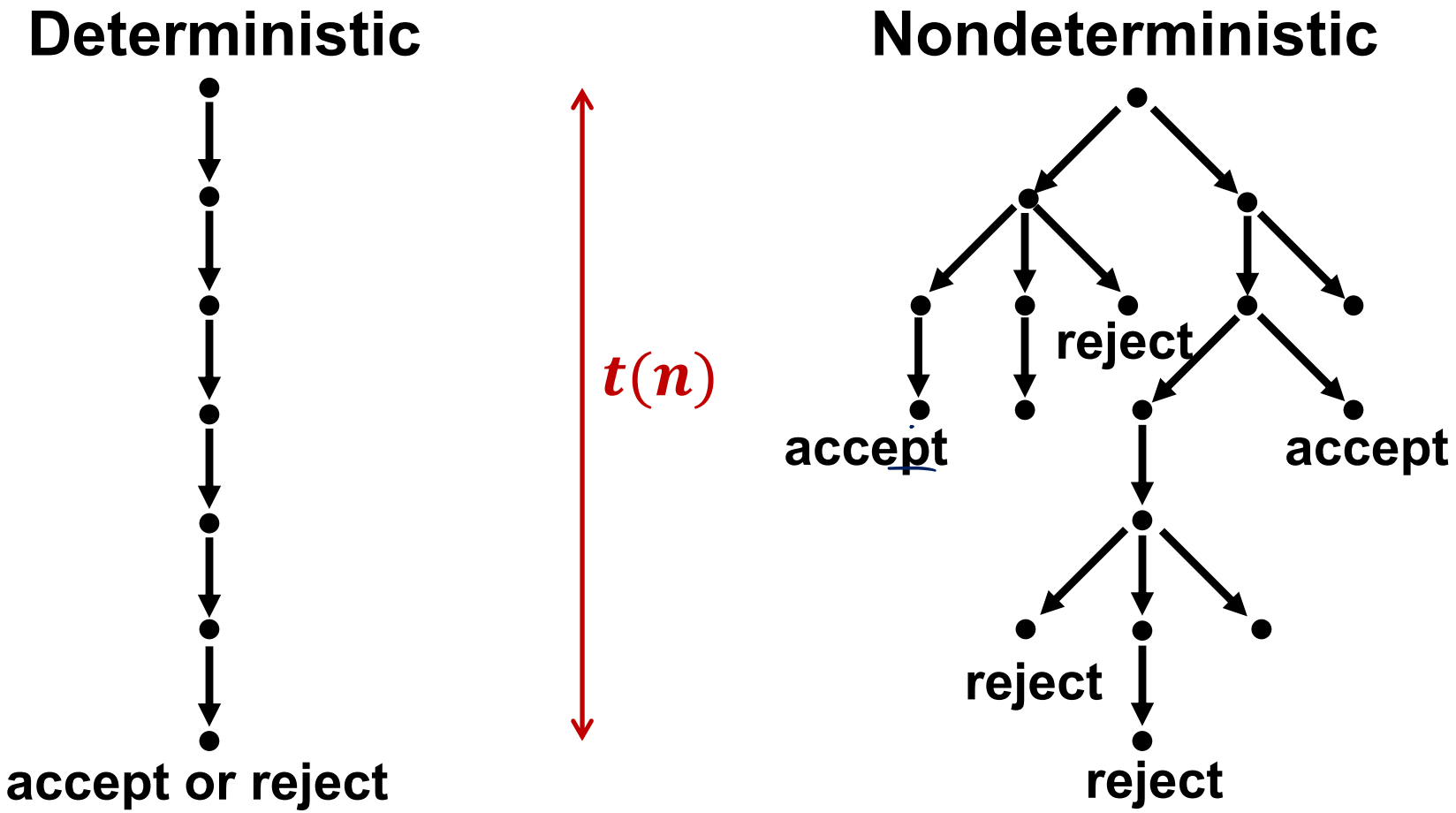
Nondeterministic time

Let t: $\mathbb{N} \rightarrow \mathbb{N}$

NTM M runs in time $t(n)$ if:

For every n and every input $w \in \Sigma^n$, M halts on w within at most $t(n)$ steps on every computational branch

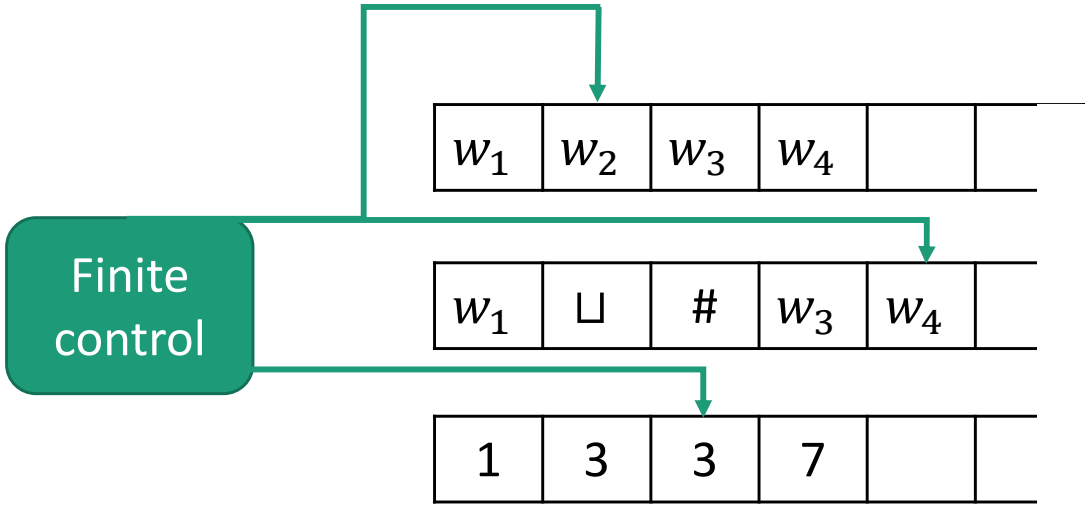
Deterministic vs. nondeterministic time



Deterministic vs. nondeterministic time

Theorem: Let $t(n) \geq n$ be a function. Every NTM running in time $t(n)$ has an equivalent deterministic single-tape TM running in time $2^{O(t(n))}$

Proof: Simulate NTM by 3-tape TM



Input w to N (read-only)

Simulation tape (run N on w using nondeterministic choices from tape 3)

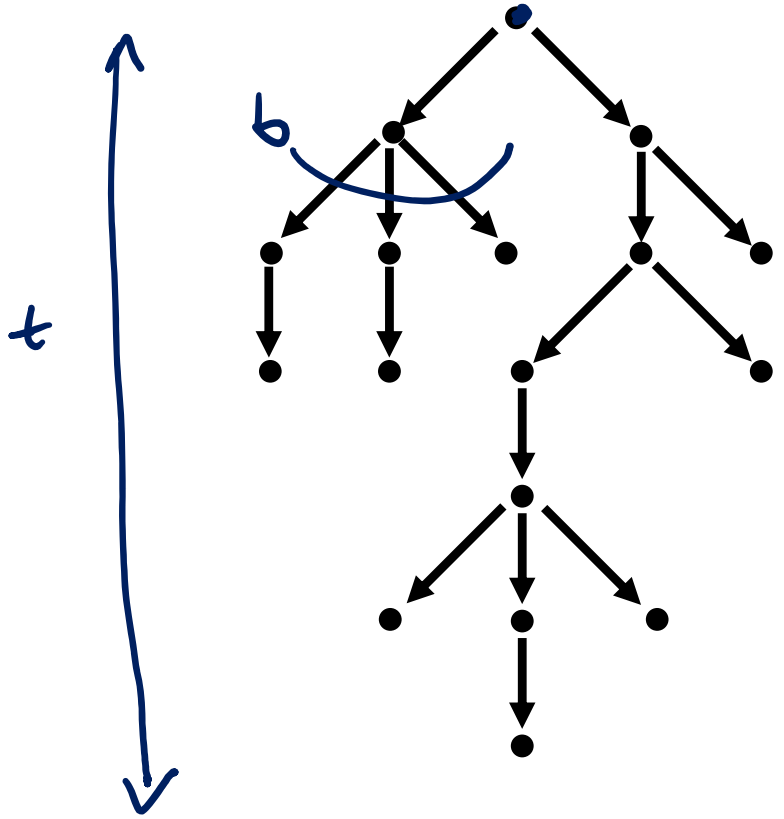
Address in computation tree

Counting leaves

What is an upper bound on the maximum number of nodes in a tree with branching factor b and depth t?

$$1 + b + b^2 + \ldots + b^{t-1}$$

$$= \frac{b^t - 1}{b - 1} \leq \boxed{b^t}$$

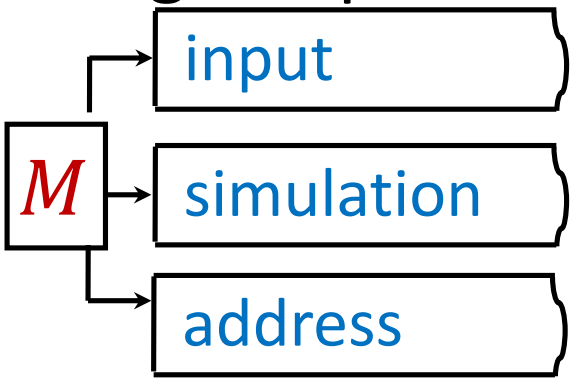


Deterministic vs. nondeterministic time

Theorem: Let $t(n) \geq n$ be a function. Every NTM running in time $t(n)$ has an equivalent deterministic single-tape TM running in time $2^{O(t(n))}$

Proof: Simulate NTM by 3-tape TM

Notice $\leq b$ choices in one step of computation



- # nodes: $b^{t(n)}$

<u>Running time:</u>

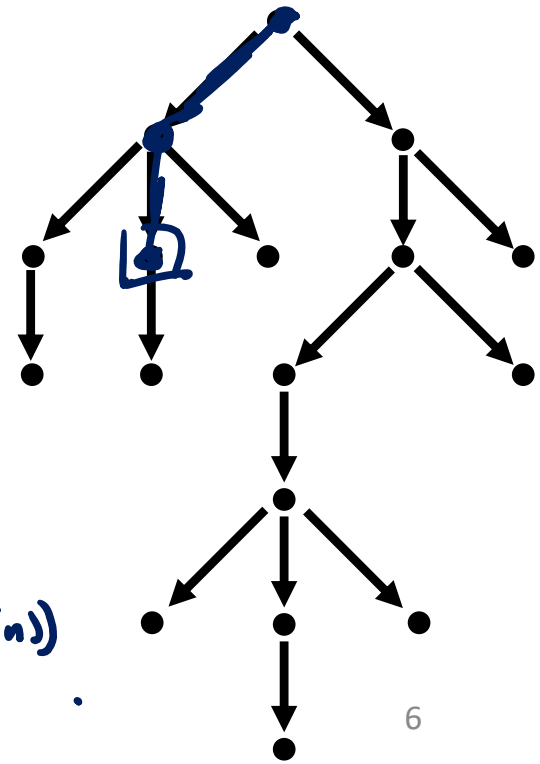
To simulate one root-to-node path: $O(t(n))$

Total time:

$\leq O(t(n)) \cdot b^{t(n)} = O(t(n)) \cdot 2^{t(n) \cdot \log b}$

$= 2^{\log(O(t(n)))} \cdot 2^{t(n) \cdot \log b}$

$= 2^{t(n) \cdot \log b + \log(O(t(n)))} \leq 2^{O(t(n))}$



Deterministic vs. nondeterministic time

Theorem: Let $t(n) \geq n$ be a function. Every NTM running in time $t(n)$ has an equivalent deterministic single-tape TM running in time $2^{O(t(n))}$

Proof: Simulate NTM by 3-tape TM in time $2^{O(t(n))}$

We know that a 3-tape TM can be simulated by a single-tape TM with quadratic overhead, hence we get running time

1-tape TM simulation overhead

$$(2^{O(t(n))})^2 = 2^{2 \cdot O(t(n))} = 2^{O(t(n))}$$

3-tape runtime

Difference in time complexity

Extended Church-Turing Thesis:

At most polynomial difference in running time between all (reasonable) deterministic models

At most exponential difference in running time between deterministic and nondeterministic models

Nondeterministic time

Let $t : \mathbb{N} \rightarrow \mathbb{N}$

NTM M runs in time $t(n)$ if:

For every n and every input $w \in \Sigma^n$, M halts on w within at most $t(n)$ steps on every computational branch

$\mathrm{NTIME}(t(n))$ is a class (i.e., set) of languages:

A language $A \in \mathrm{NTIME}(t(n))$ if there exists an NTM M that

1) Decides A, and
2) Runs in time $O(t(n))$

NTIME explicitly

A language $A \in \mathrm{NTIME}(t(n))$ if there exists an NTM M such that, on every input $w \in \Sigma^*$

1. Every computational branch of M halts in either the accept or reject state within $\mathrm{O}(t(|w|))$ steps

M runs in time O(T(n))

2. If $w \in A$, then there exists an accepting computational branch of M on input w

3. If $w \notin A$, then every computational branch of M rejects on input w

M decides A.

Complexity class NP

Definition: NP is the class of languages decidable in polynomial time on a nondeterministic TM

$$\mathrm{NP} = \bigcup_{k=1}^{\infty} \mathrm{NTIME}(n^k) = \{L \mid \exists \text{ NTM } M \text{ deciding } L \text{ in poly-time}\}$$

Which of the following are definitely true about NP?

a) $\mathrm{P} \subseteq \mathrm{NP}$

b) $\mathrm{NP} \subseteq \mathrm{P}$

c) $\mathrm{NP} \not\subseteq \mathrm{P}$

d) $\mathrm{NP} \subseteq \mathrm{EXP}$

e) $\mathrm{EXP} \subseteq \mathrm{NP}$???

$P = \{L \mid \exists$ deterministic poly-time TM deciding $L\}$

2^{n^k} for const. k

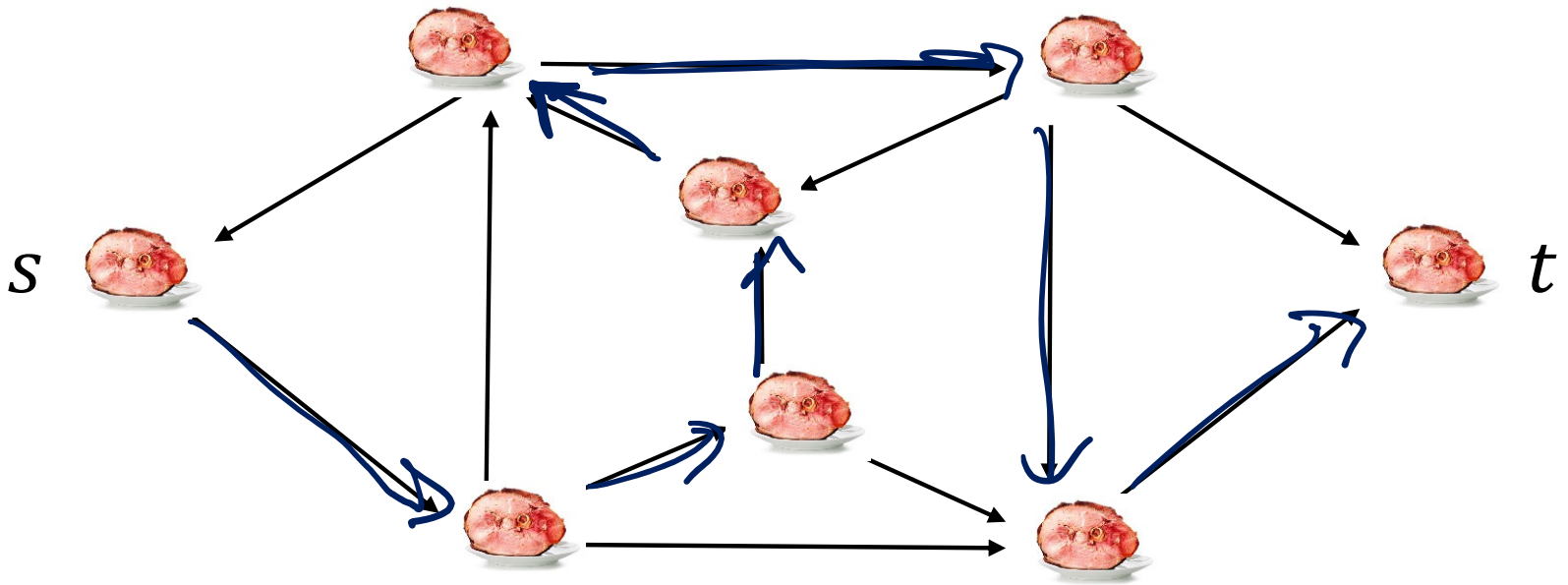
$EXP = \{L \mid \exists$ deterministic exponential-time TM deciding $L\}$

$NP = \bigcup_{k=1}^{\infty} NTIME(n^k) \subseteq \bigcup_{k=1}^{\infty} TIME(2^{O(n^k)}) = EXP$

Hamiltonian Path

$$HAMPATH = \{\langle G, s, t\rangle \mid G \text{ is a directed graph and there is a path from } s \text{ to } t \text{ that passes through every vertex exactly once}\}$$

"Hamiltonian path from s to t"



$HAMPATH \in$ NP

The following **nondeterministic** algorithm decides $HAMPATH$ in polynomial time:

On input $\langle G, s, t \rangle$: (Vertices of G are numbers $1, \ldots, k$)

1. **Nondeterministically** guess a sequence $c_1, c_2, \ldots, c_k$ of numbers $1, \ldots, k$
2. Check that $c_1, c_2, \ldots, c_k$ is a permutation: Every number $1, \ldots, k$ appears exactly once
3. Check that $c_1 = s$, $c_k = t$, and there is an edge from every c_i to c_{i+1}
4. Accept if all checks pass, otherwise, reject.

Analyzing the algorithm

On input $\langle G, s, t \rangle$:

1. **Nondeterministically** guess vertices $c_1, c_2, \ldots, c_k$
2. Check that $c_1, c_2, \ldots, c_k$ is a permutation
3. Check that $c_1 = s$, $c_k = t$, and there is an edge from every c_i to c_{i+1}
4. Accept if all checks pass, otherwise, reject.

Need to check:

1) Correctness

$\langle G,s,t \rangle \in$ HAMPATH $\Rightarrow$ $\exists$ a Hamiltonian path $c_1, \ldots, c_k$ from s to t in G
$\Rightarrow$ Branch of nondeterminism in which $c_1, \ldots, c_k$ was guessed leads all checks to pass $\Rightarrow$ NTM accepts on this branch

$\langle G,s,t \rangle \notin$ HAMPATH $\Rightarrow$ there is no Hamiltonian path from s to t
$\Rightarrow$ every vertex sequence $c_1, \ldots, c_k$ either fails to be an $s \to t$ path or fails to hit every vertex once
$\Rightarrow$ every branch of computation leads to reject.

2) Running time

1. Each vertex takes $O(\log k)$ bits to describe $\Rightarrow$ $O(k \log k)$ time to guess
2. Count $i = 1, \ldots, k$ and check each i appears among $c_1, \ldots, c_k$
$k \cdot O((k \log k)^2) = O(k^3 \log^2 k)$
3. k pairs of vertices to check
$O((k \log k)^2)$ time to look up each pair in adjacency matrix
$\Big\}$ $O(k^3 \log^2 k)$ time

Total runtime: $O(k^3 \log^2 k)$

Nondeterministically guessing, then checking

How did we design an NTM for HAMPATH?

- Given a candidate path, it is easy (poly-time) to check whether this path is a Hamiltonian path
- We designed a poly-time NTM by nondeterministically guessing this path and then deterministically checking it
- Lots of problems have this structure (CLIQUE, 3-COLOR, FACTOR,…). They might be hard to solve, but a candidate solution is easy to check.

General structure: $w \in L$ if and only if there exists a nondeterministically guessable, but deterministically checkable "certificate" c to that fact

An alternative characterization of NP

"Languages with polynomial-time verifiers"

A verifier for a language L is a deterministic algorithm V such that $w \in L$ iff there exists a string c such that $V(\langle w, c \rangle)$ accepts

"certificate" "witness" "proof"

Running time of a verifier is only measured in terms of $|w|$

V is a polynomial-time verifier if it runs in time polynomial in $|w|$ on every input $\langle w, c \rangle$

(Without loss of generality, $|c|$ is polynomial in $|w|$, i.e., $|c| = O(|w|^k)$ for some constant k)

$HAMPATH$ has a polynomial-time verifier

Certificate c: $c_1, \ldots, c_n$ representing an alleged Hamiltonian path

Verifier V: instance w certificate

On input $\langle G, s, t; c \rangle$: (Vertices of G are numbers $1, \ldots, k$)

1. Check that $c_1, c_2, \ldots, c_k$ is a permutation: Every number $1, \ldots, k$ appears exactly once
2. Check that $c_1 = s$, $c_k = t$, and there is an edge from every c_i to c_{i+1}
3. Accept if all checks pass, otherwise, reject.

Correctness: $\langle G,s,t \rangle \in HAMPATH \Rightarrow \exists$ a Hamiltonian path $c_1, \ldots, c_n$ from $s \to t$ in G

$\Rightarrow V$ accepts on input $\langle w, c \rangle$

$\langle G,s,t \rangle \notin HAMPATH \Rightarrow$ there is no Hamiltonian path

$\Rightarrow \forall c$, V on $\langle w, c \rangle$ rejects

Runtime: Same routine proof as for NTM, plus fact that $|c| = O(n \log n)$ is polynomial in $|\langle G,s,t \rangle|$

NP is the class of languages with polynomial-time verifiers

Theorem: A language $L \in \mathrm{NP}$ iff there is a polynomial-time verifier for L

Alternative proof of NP ⊆ EXP

$w \in L \Rightarrow \exists\ c$ s.t. $V(\langle w, c\rangle)$ accepts

$w \notin L \Rightarrow \forall\ c$ $V(\langle w, c\rangle)$ rejects

One can prove NP ⊆ EXP as follows. Let V be a verifier for an NP language L running in time $T(n)$. We can construct a deterministic $2^{O(T(n))}$ time algorithm M deciding L as follows.

some polynomial

a) On input $\langle w, c\rangle$, run V on $\langle w, c\rangle$ and output the result

b) On input w, run V on all possible $\langle w, c\rangle$, where c is a certificate string. Accept if any run accepts.

c) On input w, run V on all possible $\langle w, c\rangle$, where c is a certificate of length at most $T(|w|)$. Accept if any run accepts.

runtime $2^{O(T(|w|))} \cdot T(|w|) = 2^{O(T(|w|))}$

d) On input w, run V on all possible $\langle x, c\rangle$, where x is a string of length $|w|$ and c is a certificate of length at most $T(|w|)$. Accept if any run accepts.

NP is the class of languages with polynomial-time verifiers

Theorem: A language $L \in \text{NP}$ iff there is a polynomial-time verifier for L

Proof: $\Leftarrow$ Let L have a time-$T(n)$ verifier $V(\langle w, c \rangle)$ (some polynomial)

Idea: Design NTM N for L that nondeterministically guesses a certificate

Runtime

On input w:

1. Nondeterministically guess c of length $\leq T(|w|)$ — $O(T(n))$
2. Run V on input $\langle w, c \rangle$. If accepts, accept. If reject, reject. — $O(T(n))$

Correctness: If $w \in L$: $\exists\, c$ s.t. $V(\langle w, c \rangle)$ accepts
$\Rightarrow$ Branch where c is guessed leads N to accept.

If $w \notin L$: $\forall\, c$ causes $V(\langle w, c \rangle)$ to reject.
$\Rightarrow$ Every branch leads N to reject.