Lesson 9. Non-deterministic Turing machines

CSIE 3110 – Formal Languages and Automata Theory

Tony Tan
Department of Computer Science and Information Engineering
College of Electrical Engineering and Computer Science
National Taiwan University

Table of contents

Table of contents

Non-deterministic Turing machines

We have learnt that Turing machines are the formal definition of algorithms.

Non-deterministic Turing machines

We have learnt that Turing machines are the formal definition of algorithms.

In this lesson we will discuss non-deterministic Turing machines.

Non-deterministic Turing machines

We have learnt that Turing machines are the formal definition of algorithms.

In this lesson we will discuss non-deterministic Turing machines.

Intuitively, non-deterministic Turing machines are algorithms with capability to "guess correctly."

Non-deterministic Turing machines

We have learnt that Turing machines are the formal definition of algorithms.

In this lesson we will discuss non-deterministic Turing machines.

Intuitively, non-deterministic Turing machines are algorithms with capability to "guess correctly."

They are an important model of computation for defining complexity classes such as the class NP-complete.

The definition of non-deterministic Turing machine

(Def.) A *non-deterministic Turing machine* (NTM) is:

$$\mathcal{M} = \langle \Sigma, \Gamma, Q, q_0, q_{\mathrm{acc}}, q_{\mathrm{rej}}, \delta \rangle$$

All the components are defined as in the standard Turing machine.

The difference is that δ is now a relation, where there is one or two transitions applicable on every pair $(p, a) \in Q \times \Gamma$.

The definition of non-deterministic Turing machine

(Def.) A *non-deterministic Turing machine* (NTM) is:

$$\mathcal{M} = \langle \Sigma, \Gamma, Q, q_0, q_{\mathrm{acc}}, q_{\mathrm{rej}}, \delta \rangle$$

All the components are defined as in the standard Turing machine.

The difference is that δ is now a relation, where there is one or two transitions applicable on every pair $(p, a) \in Q \times \Gamma$.

More precisely, for every pair $(p, a) \in Q \times \Gamma$:

Either there is exactly *one* (q, b, α) such that:

$$(p, a) \to (q, b, \alpha) \ \in \ \delta$$

or there are exactly *two* (q_1, b_1, α_1) *and* (q_2, b_2, α_2) such that:

$$(p, a) \to (q_1, b_1, \alpha_1) \ \in \ \delta \qquad \text{and} \qquad (p, a) \to (q_2, b_2, \alpha_2) \ \in \ \delta$$

Some remarks

In the standard Turing machine, there is exactly one transition applicable on every pair $(p, a) \in Q \times \Gamma$.

Some remarks

In the standard Turing machine, there is exactly one transition applicable on every pair $(p, a) \in Q \times \Gamma$.

It works "deterministically":

For every (p, a), there is only one (q, b, α) such that $(p, a) \to (q, b, \alpha) \in \delta$.

Some remarks

In the standard Turing machine, there is exactly one transition applicable on every pair $(p, a) \in Q \times \Gamma$.

It works "deterministically":

For every (p, a), there is only one (q, b, α) such that $(p, a) \to (q, b, \alpha) \in \delta$.

It is usually called deterministic Turing machine (DTM).

NTM vs. DTM

NTM vs. DTM $\equiv$ NFA vs. DFA

NTM vs. DTM

NTM vs. DTM $\equiv$ NFA vs. DFA

The notions of configuration, initial configuration, accepting/rejecting configuration and run for NTM are all defined exactly as in DTM.

NTM vs. DTM

NTM vs. DTM $\equiv$ NFA vs. DFA

The notions of configuration, initial configuration, accepting/rejecting configuration and run for NTM are all defined exactly as in DTM.

- For every input word w, there is exactly one run of a DTM on w.
- For every input word w, there are many runs of an NTM on w.

Illustration

Let $\mathcal{M}$ be an NTM and w be the input word.

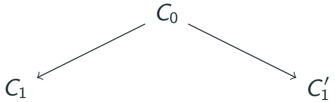
Illustration

Let $\mathcal{M}$ be an NTM and w be the input word.

$$C_0$$

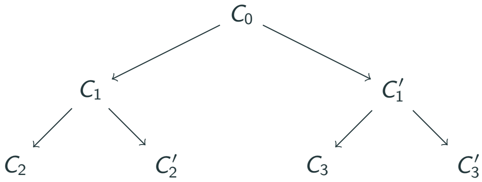
Illustration

Let $\mathcal{M}$ be an NTM and w be the input word.



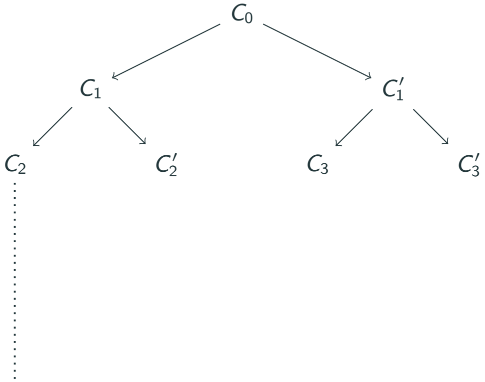
Illustration

Let $\mathcal{M}$ be an NTM and w be the input word.



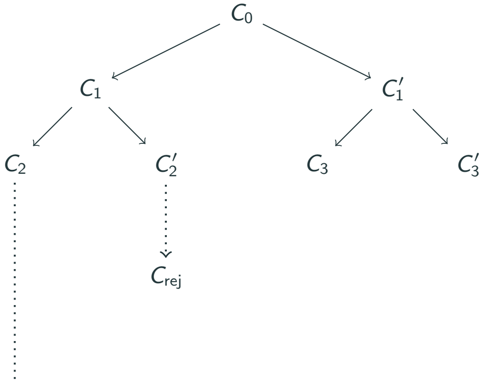
Illustration

Let $\mathcal{M}$ be an NTM and w be the input word.



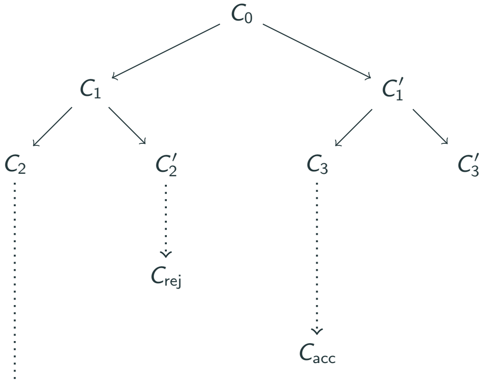
Illustration

Let $\mathcal{M}$ be an NTM and w be the input word.



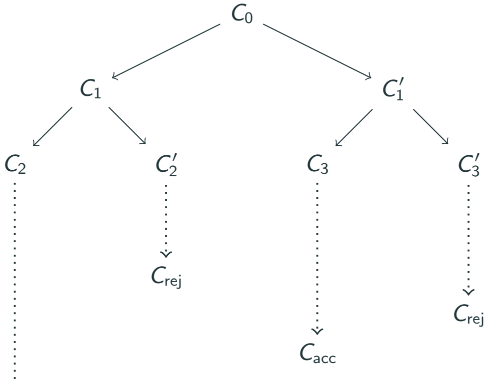
Illustration

Let $\mathcal{M}$ be an NTM and w be the input word.



Illustration

Let $\mathcal{M}$ be an NTM and w be the input word.



The acceptance condition of an NTM

(Def.) An NTM $\mathcal{M}$ *accepts w*, if there is an accepting run of $\mathcal{M}$ on w.

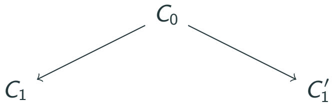
The acceptance condition of an NTM

(Def.) An NTM $\mathcal{M}$ *accepts* w, if there is an accepting run of $\mathcal{M}$ on w.

$$C_0$$

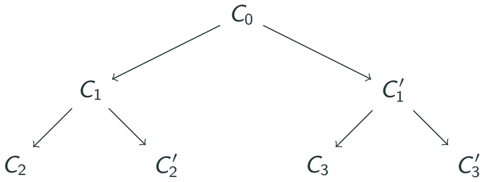
The acceptance condition of an NTM

(Def.) An NTM $\mathcal{M}$ *accepts w*, if there is an accepting run of $\mathcal{M}$ on w.



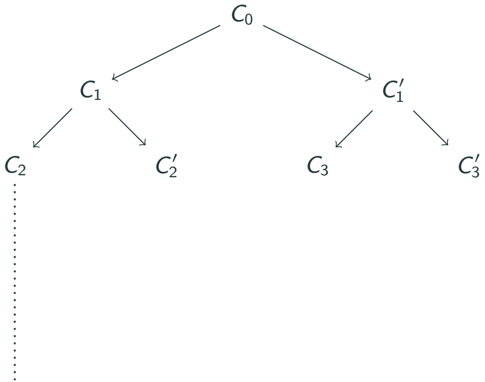
The acceptance condition of an NTM

(Def.) An NTM $\mathcal{M}$ *accepts w*, if there is an accepting run of $\mathcal{M}$ on w.



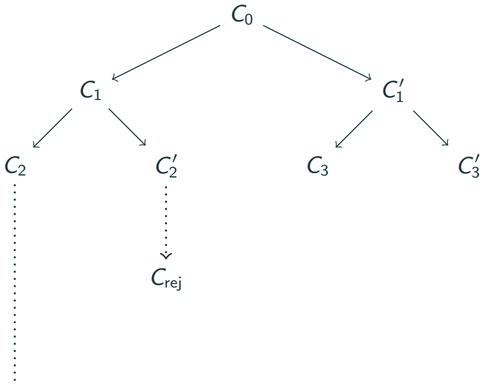
The acceptance condition of an NTM

(Def.) An NTM *$\mathcal{M}$ accepts w*, if there is an accepting run of $\mathcal{M}$ on w.



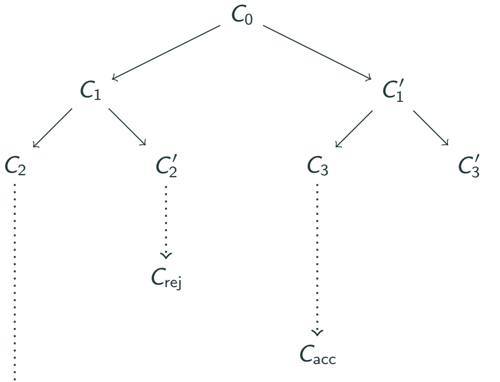
The acceptance condition of an NTM

(Def.) An NTM $\mathcal{M}$ *accepts w*, if there is an accepting run of $\mathcal{M}$ on w.



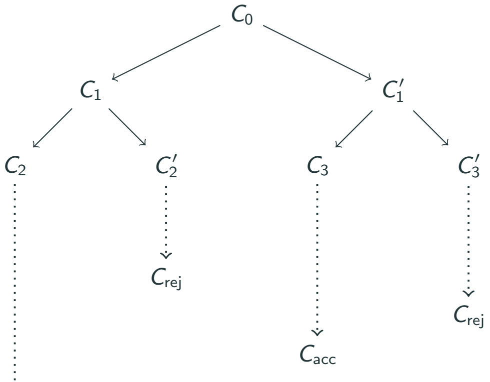
The acceptance condition of an NTM

(Def.) An NTM $\mathcal{M}$ *accepts w*, if there is an accepting run of $\mathcal{M}$ on w.



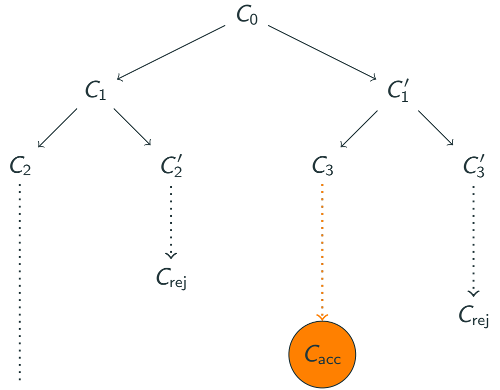
The acceptance condition of an NTM

(Def.) An NTM $\mathcal{M}$ *accepts* w, if there is an accepting run of $\mathcal{M}$ on w.



The acceptance condition of an NTM

(Def.) An NTM $\mathcal{M}$ *accepts w*, if there is an accepting run of $\mathcal{M}$ on w.



Rejection condition of an NTM

(Def.) An NTM $\mathcal{M}$ *rejects* w, if all the runs of $\mathcal{M}$ on w are rejecting.

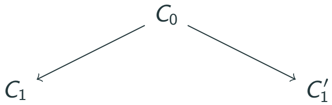
Rejection condition of an NTM

(Def.) An NTM $\mathcal{M}$ *rejects w*, if all the runs of $\mathcal{M}$ on w are rejecting.

$$C_0$$

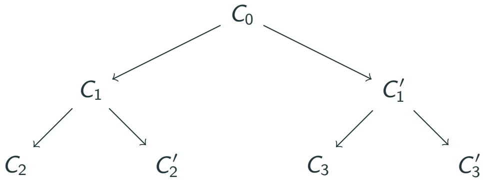
Rejection condition of an NTM

(Def.) An NTM $\mathcal{M}$ *rejects w*, if all the runs of $\mathcal{M}$ on w are rejecting.



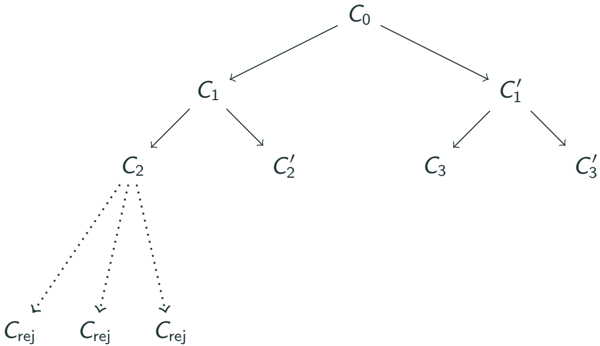
Rejection condition of an NTM

(Def.) An NTM $\mathcal{M}$ *rejects w*, if all the runs of $\mathcal{M}$ on w are rejecting.



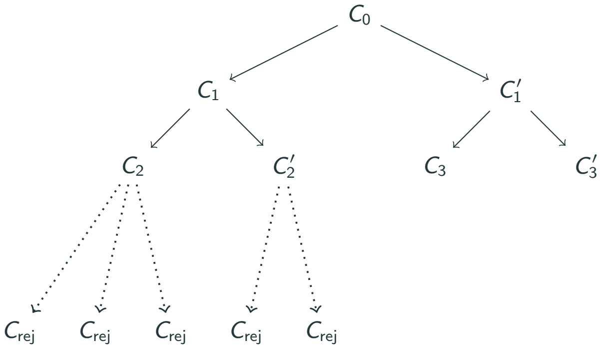
Rejection condition of an NTM

(Def.) An NTM $\mathcal{M}$ *rejects* w, if all the runs of $\mathcal{M}$ on w are rejecting.



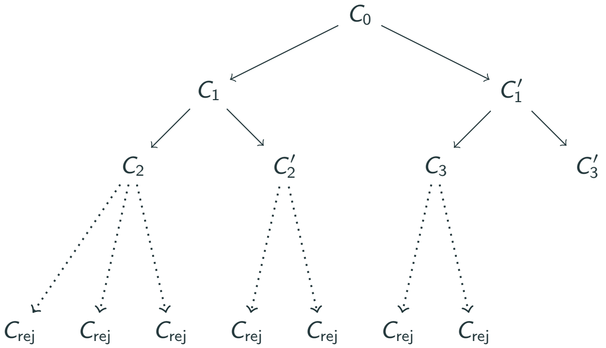
Rejection condition of an NTM

(Def.) An NTM $\mathcal{M}$ *rejects* w, if all the runs of $\mathcal{M}$ on w are rejecting.



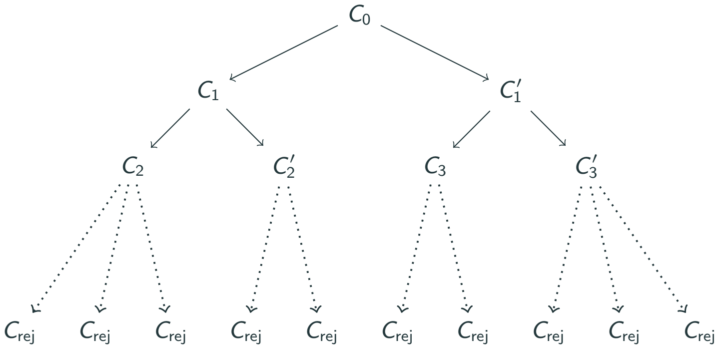
Rejection condition of an NTM

(Def.) An NTM $\mathcal{M}$ *rejects w*, if all the runs of $\mathcal{M}$ on w are rejecting.



Rejection condition of an NTM

(Def.) An NTM $\mathcal{M}$ *rejects w*, if all the runs of $\mathcal{M}$ on w are rejecting.



When NTM does not halt

An NTM $\mathcal{M}$ *does not halt on w*, if $\mathcal{M}$ neither accept nor reject w, i.e., $\mathcal{M}$ does not accept w and it also does not reject w.

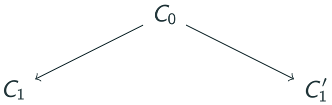
When NTM does not halt

An NTM *$\mathcal{M}$ does not halt on w*, if $\mathcal{M}$ neither accept nor reject w, i.e., $\mathcal{M}$ does not accept w and it also does not reject w.

$$C_0$$

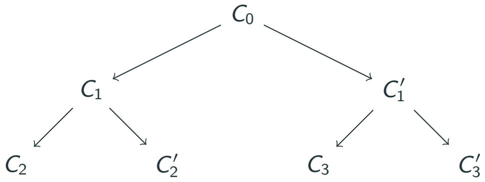
When NTM does not halt

An NTM $\mathcal{M}$ *does not halt on w*, if $\mathcal{M}$ neither accept nor reject w, i.e., $\mathcal{M}$ does not accept w and it also does not reject w.



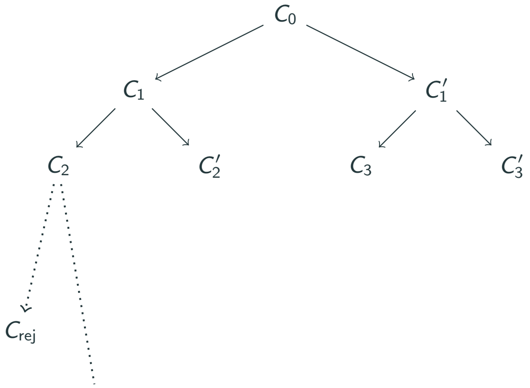
When NTM does not halt

An NTM $\mathcal{M}$ *does not halt on w*, if $\mathcal{M}$ neither accept nor reject w, i.e., $\mathcal{M}$ does not accept w and it also does not reject w.



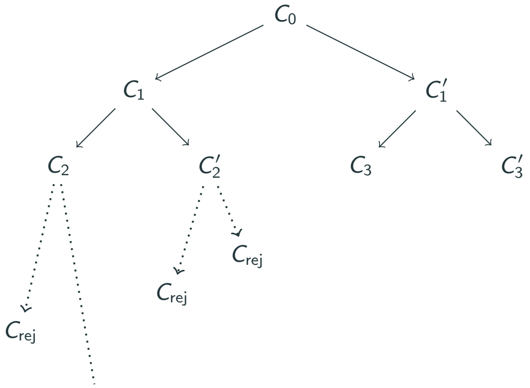
When NTM does not halt

An NTM $\mathcal{M}$ *does not halt on w*, if $\mathcal{M}$ neither accept nor reject w, i.e., $\mathcal{M}$ does not accept w and it also does not reject w.



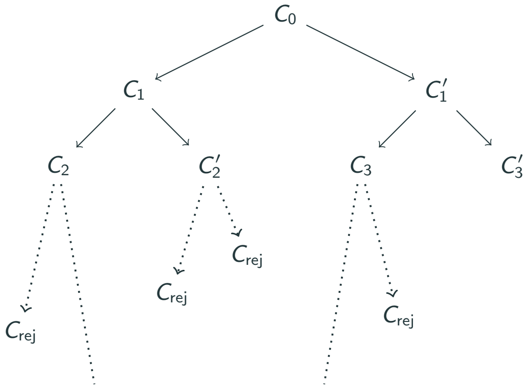
When NTM does not halt

An NTM $\mathcal{M}$ *does not halt on w*, if $\mathcal{M}$ neither accept nor reject w, i.e., $\mathcal{M}$ does not accept w and it also does not reject w.



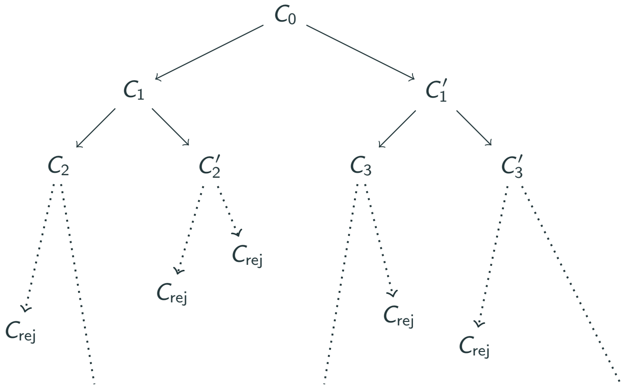
When NTM does not halt

An NTM $\mathcal{M}$ *does not halt on w*, if $\mathcal{M}$ neither accept nor reject w, i.e., $\mathcal{M}$ does not accept w and it also does not reject w.



When NTM does not halt

An NTM $\mathcal{M}$ *does not halt on w*, if $\mathcal{M}$ neither accept nor reject w, i.e., $\mathcal{M}$ does not accept w and it also does not reject w.



Decidable and recognizable languages by NTM

(Def.) An NTM $\mathcal{M}$ decides a language L, if:

- for every $w \in L$, $\mathcal{M}$ accepts w;
- for every $w \notin L$, $\mathcal{M}$ rejects w.

Decidable and recognizable languages by NTM

(Def.) An NTM $\mathcal{M}$ decides a language L, if:

- for every $w \in L$, $\mathcal{M}$ accepts w;
- for every $w \notin L$, $\mathcal{M}$ rejects w.

(Def.) An NTM $\mathcal{M}$ recognizes a language L, if:

- for every $w \in L$, $\mathcal{M}$ accepts w;
- for every $w \notin L$, $\mathcal{M}$ does not accept w.

NTM is equivalent to DTM

Theorem 9.1

For every language NTM $\mathcal{M}$, there is DTM $\mathcal{M}'$ such that for every input word w, the following holds.

- *If $\mathcal{M}$ accepts w, then $\mathcal{M}'$ accepts w.*
- *If $\mathcal{M}$ rejects w, then $\mathcal{M}'$ rejects w.*
- *If $\mathcal{M}$ does not halt on w, then $\mathcal{M}'$ does not halt on w.*

In other words, $\mathcal{M}$ and $\mathcal{M}'$ are equivalent.

Proof of Theorem 9.1

Let $\mathcal{M}$ be an NTM.

The DTM $\mathcal{M}'$ works by simulating $\mathcal{M}$ on the input word.

On input word w, do the following.

- Let C_0 be the initial configuration of $\mathcal{M}$ on w.
- Let $S = \{C_0\}$, i.e., a set that contains only one element C_0.
- while ($S \neq \emptyset$) or (S contains an accepting configuration):
  - Delete all the rejecting configurations from S.
  - Compute the next configuration of each element in S.
    Store them all in S.
- If S contains an accepting configuration, ACCEPT.
  If $S = \emptyset$, REJECT.

Proof of Theorem 9.1 – Illustration

On input word w:

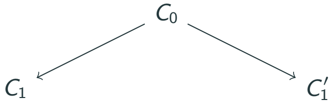
Proof of Theorem 9.1 – Illustration

On input word w:

$$C_0$$

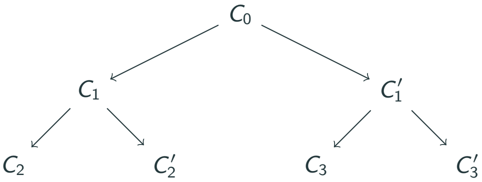
Proof of Theorem 9.1 – Illustration

On input word w:



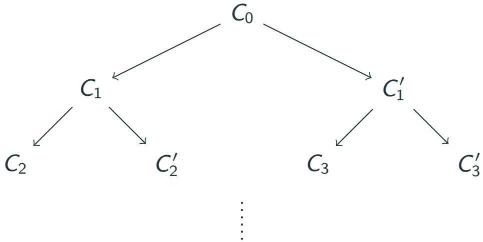
Proof of Theorem 9.1 – Illustration

On input word w:



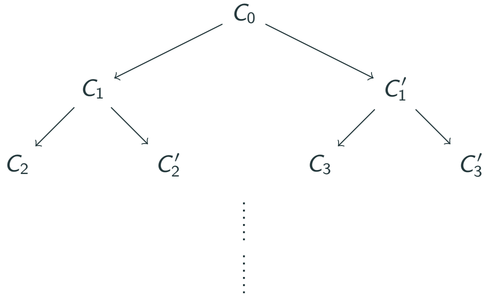
Proof of Theorem 9.1 – Illustration

On input word w:



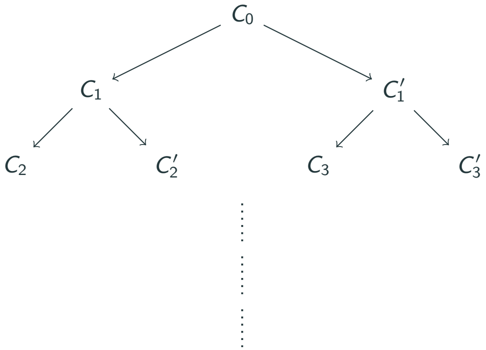
Proof of Theorem 9.1 – Illustration

On input word w:



Proof of Theorem 9.1 – Illustration

On input word w:



NTM is equivalent to DTM

Theorem 9.1

For every language NTM $\mathcal{M}$, there is DTM $\mathcal{M}'$ such that for every input word w, the following holds.

- *If $\mathcal{M}$ accepts w, then $\mathcal{M}'$ accepts w.*
- *If $\mathcal{M}$ rejects w, then $\mathcal{M}'$ rejects w.*
- *If $\mathcal{M}$ does not halt on w, then $\mathcal{M}'$ does not halt on w.*

In other words, $\mathcal{M}$ and $\mathcal{M}'$ are equivalent.

NTM is equivalent to DTM

Theorem 9.1

For every language NTM $\mathcal{M}$, there is DTM $\mathcal{M}'$ such that for every input word w, the following holds.

- *If $\mathcal{M}$ accepts w, then $\mathcal{M}'$ accepts w.*
- *If $\mathcal{M}$ rejects w, then $\mathcal{M}'$ rejects w.*
- *If $\mathcal{M}$ does not halt on w, then $\mathcal{M}'$ does not halt on w.*

In other words, $\mathcal{M}$ and $\mathcal{M}'$ are equivalent.

NTM can be generalized to multiple tape and Theorem 9.1 still holds.

Closure property of recognizable languages

Theorem 9.2

Recognizable languages are closed under concatenation and Kleene star.

Closure property of recognizable languages

Theorem 9.2
Recognizable languages are closed under concatenation and Kleene star.

(Proof) Let L_1 and L_2 be recognizable languages and let $\mathcal{M}_1$ and $\mathcal{M}_2$ be DTM that recognize them. We assume that $\Sigma = \{0, 1\}$.

Closure property of recognizable languages

Theorem 9.2

Recognizable languages are closed under concatenation and Kleene star.

(Proof) Let L_1 and L_2 be recognizable languages and let $\mathcal{M}_1$ and $\mathcal{M}_2$ be DTM that recognize them. We assume that $\Sigma = \{0, 1\}$.

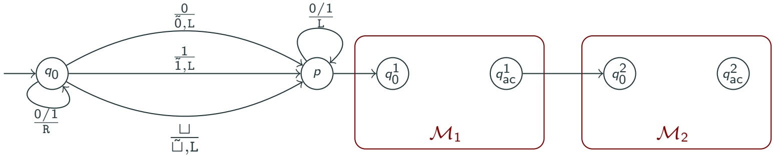
(Closure under concatenation) We present a 2-tape NTM $\mathcal{M}$ that recognizes $L_1 L_2$. On input word w:

- "Guess" a partition $v_1 v_2$ of w.
- Copy v_1 onto the second tape.
- Run $\mathcal{M}_1$ on v_1 (on the second tape).
- If $\mathcal{M}_1$ accepts, erase the second tape and copy v_2 onto the second tape
- Run $\mathcal{M}_2$ on v_2 (on the second tape).
- If $\mathcal{M}_2$ accepts, ACCEPT.

“Guess” a partition of w into $w = v_1 v_2$ – Illustration

We have new symbols $\tilde{0}, \tilde{1}, \tilde{\sqcup}$.

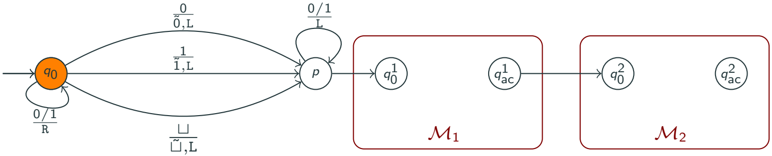
The NTM looks like this:



“Guess” a partition of w into $w = v_1 v_2$ – Illustration

We have new symbols $\tilde{0}, \tilde{1}, \tilde{\sqcup}$.

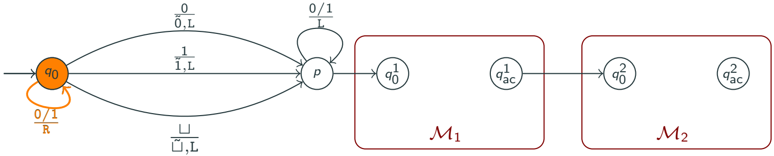
The NTM looks like this:



"Guess" a partition of w into $w = v_1 v_2$ – Illustration

We have new symbols $\tilde{0}, \tilde{1}, \tilde{\sqcup}$.

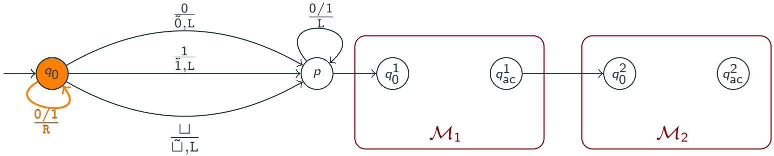
The NTM looks like this:



“Guess” a partition of w into $w = v_1 v_2$ – Illustration

We have new symbols $\tilde{0}, \tilde{1}, \tilde{\sqcup}$.

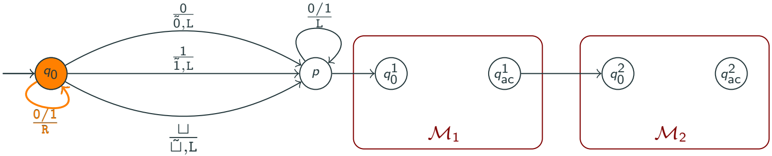
The NTM looks like this:



“Guess” a partition of w into $w = v_1 v_2$ – Illustration

We have new symbols $\tilde{0}, \tilde{1}, \tilde{\sqcup}$.

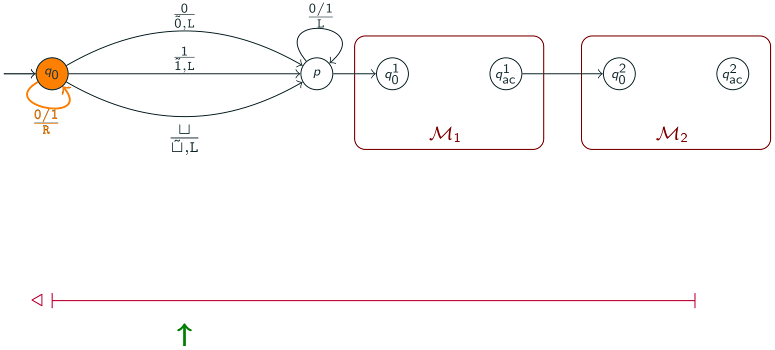
The NTM looks like this:



“Guess” a partition of w into $w = v_1 v_2$ – Illustration

We have new symbols $\tilde{0}, \tilde{1}, \tilde{\sqcup}$.

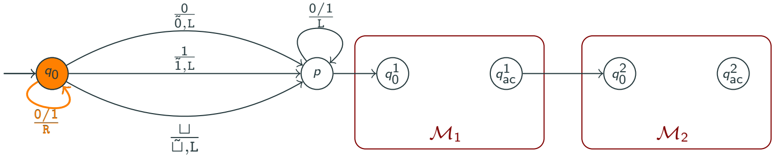
The NTM looks like this:



“Guess” a partition of w into $w = v_1 v_2$ – Illustration

We have new symbols $\tilde{0}, \tilde{1}, \tilde{\sqcup}$.

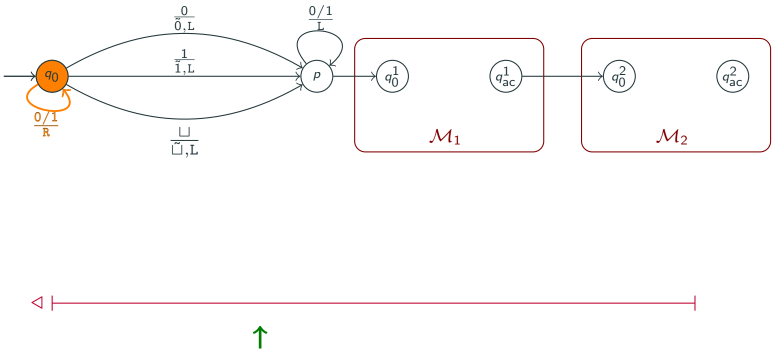
The NTM looks like this:



“Guess” a partition of w into $w = v_1 v_2$ – Illustration

We have new symbols $\tilde{0}, \tilde{1}, \tilde{\sqcup}$.

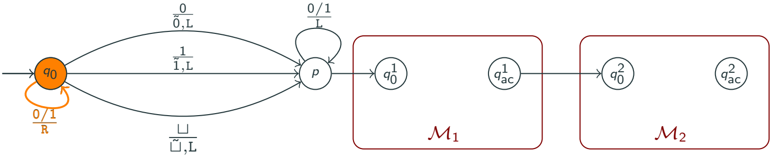
The NTM looks like this:



“Guess” a partition of w into $w = v_1 v_2$ – Illustration

We have new symbols $\tilde{0}, \tilde{1}, \tilde{\sqcup}$.

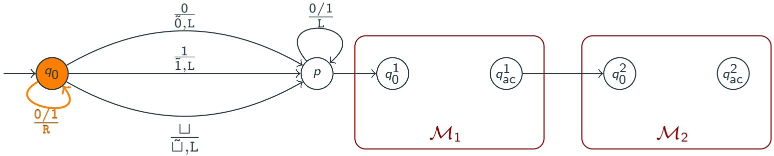
The NTM looks like this:



“Guess” a partition of w into $w = v_1 v_2$ – Illustration

We have new symbols $\tilde{0}, \tilde{1}, \tilde{\sqcup}$.

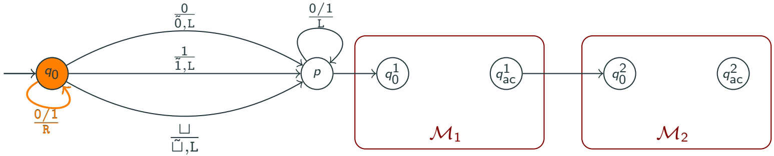
The NTM looks like this:



“Guess” a partition of w into $w = v_1 v_2$ – Illustration

We have new symbols $\tilde{0}, \tilde{1}, \tilde{\sqcup}$.

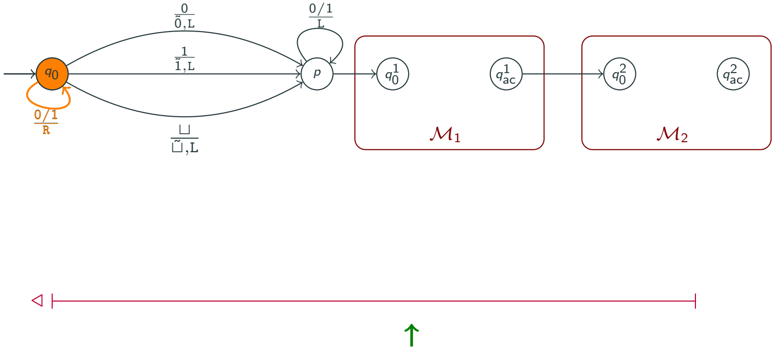
The NTM looks like this:



"Guess" a partition of w into $w = v_1 v_2$ – Illustration

We have new symbols $\tilde{0}, \tilde{1}, \tilde{\sqcup}$.

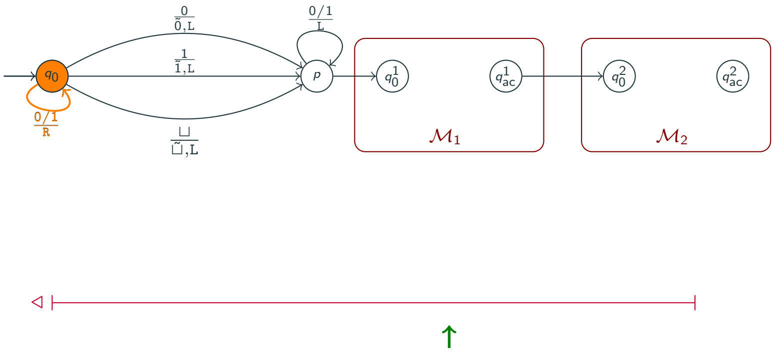
The NTM looks like this:



“Guess” a partition of w into $w = v_1 v_2$ – Illustration

We have new symbols $\tilde{0}, \tilde{1}, \tilde{\sqcup}$.

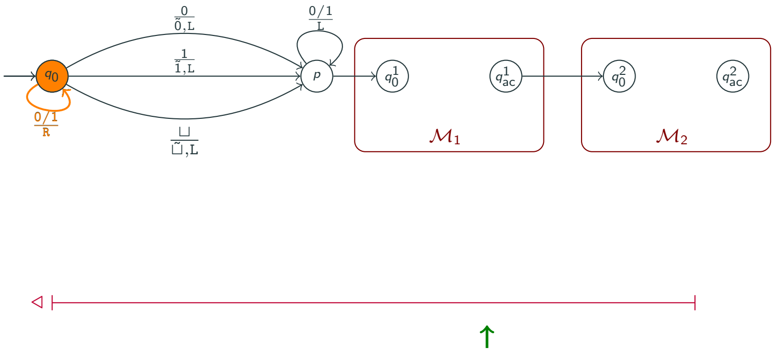
The NTM looks like this:



“Guess” a partition of w into $w = v_1 v_2$ – Illustration

We have new symbols $\tilde{0}, \tilde{1}, \tilde{\sqcup}$.

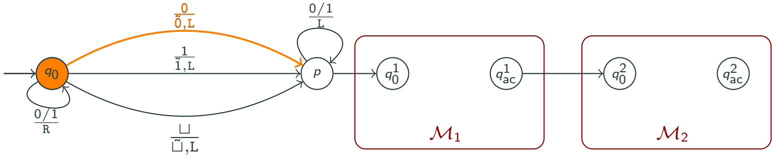
The NTM looks like this:



“Guess” a partition of w into $w = v_1 v_2$ – Illustration

We have new symbols $\tilde{0}, \tilde{1}, \tilde{\sqcup}$.

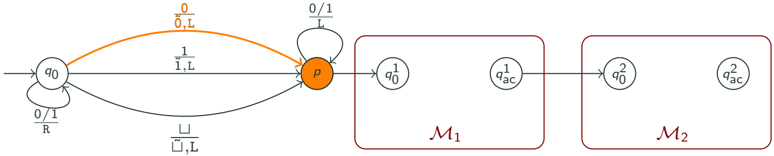
The NTM looks like this:



“Guess” a partition of w into $w = v_1 v_2$ – Illustration

We have new symbols $\tilde{0}, \tilde{1}, \tilde{\sqcup}$.

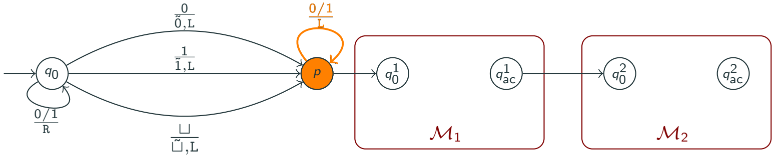
The NTM looks like this:



“Guess” a partition of w into $w = v_1 v_2$ – Illustration

We have new symbols $\tilde{0}, \tilde{1}, \tilde{\sqcup}$.

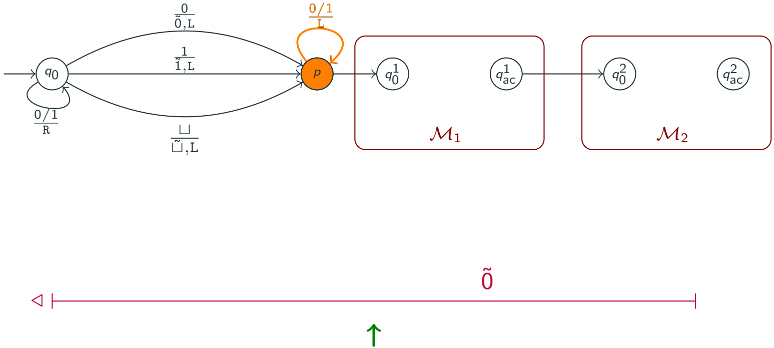
The NTM looks like this:



"Guess" a partition of w into $w = v_1 v_2$ – Illustration

We have new symbols $\tilde{0}, \tilde{1}, \tilde{\sqcup}$.

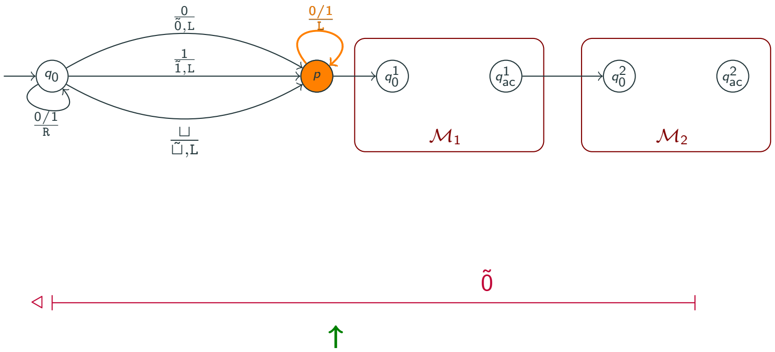
The NTM looks like this:



“Guess” a partition of w into $w = v_1 v_2$ – Illustration

We have new symbols $\tilde{0}, \tilde{1}, \tilde{\sqcup}$.

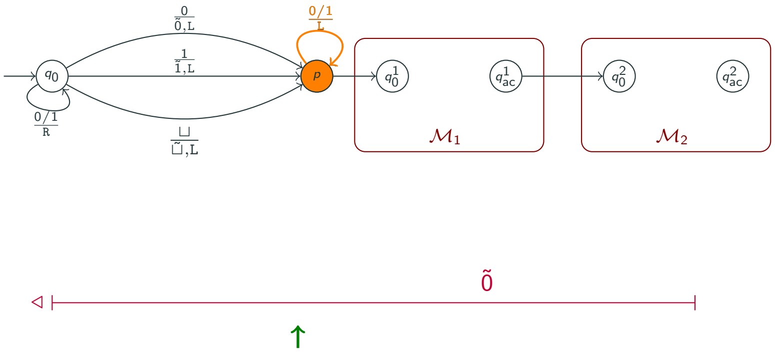
The NTM looks like this:



“Guess” a partition of w into $w = v_1 v_2$ – Illustration

We have new symbols $\tilde{0}, \tilde{1}, \tilde{\sqcup}$.

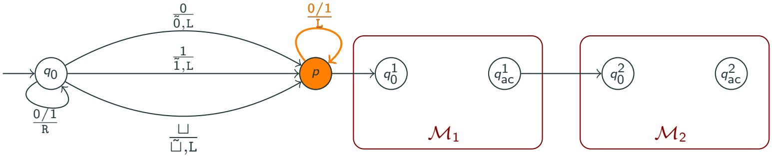
The NTM looks like this:



“Guess” a partition of w into $w = v_1 v_2$ – Illustration

We have new symbols $\tilde{0}, \tilde{1}, \tilde{\sqcup}$.

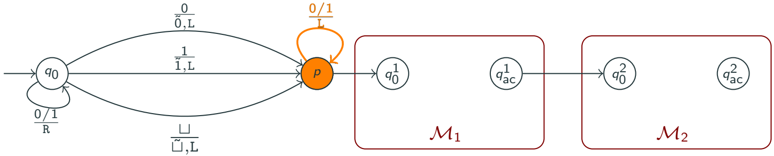
The NTM looks like this:



“Guess” a partition of w into $w = v_1 v_2$ – Illustration

We have new symbols $\tilde{0}, \tilde{1}, \tilde{\sqcup}$.

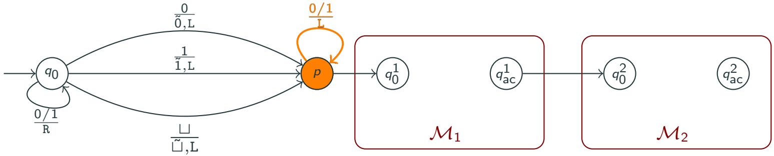
The NTM looks like this:



“Guess” a partition of w into $w = v_1 v_2$ – Illustration

We have new symbols $\tilde{0}, \tilde{1}, \tilde{\sqcup}$.

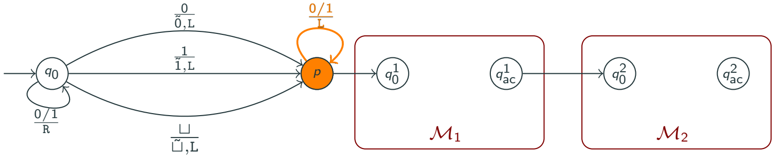
The NTM looks like this:



“Guess” a partition of w into $w = v_1 v_2$ – Illustration

We have new symbols $\tilde{0}, \tilde{1}, \tilde{\sqcup}$.

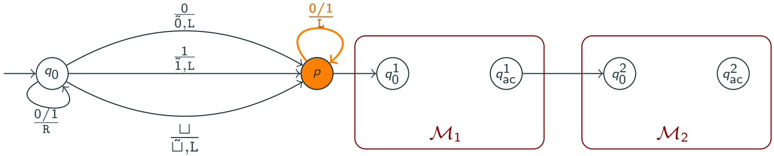
The NTM looks like this:



“Guess” a partition of w into $w = v_1 v_2$ – Illustration

We have new symbols $\tilde{0}, \tilde{1}, \tilde{\sqcup}$.

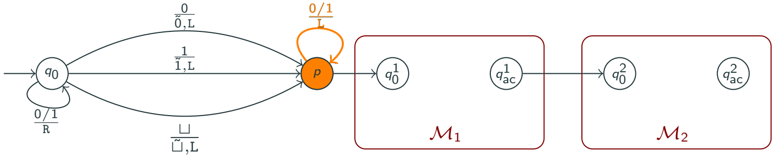
The NTM looks like this:



“Guess” a partition of w into $w = v_1 v_2$ – Illustration

We have new symbols $\tilde{0}, \tilde{1}, \tilde{\sqcup}$.

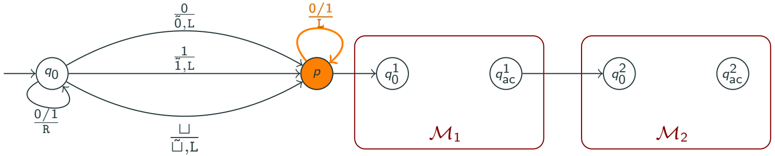
The NTM looks like this:



“Guess” a partition of w into $w = v_1 v_2$ – Illustration

We have new symbols $\tilde{0}, \tilde{1}, \tilde{\sqcup}$.

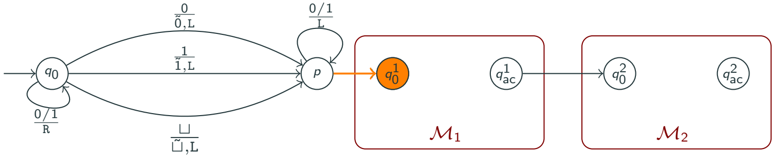
The NTM looks like this:



"Guess" a partition of w into $w = v_1 v_2$ – Illustration

We have new symbols $\tilde{0}, \tilde{1}, \tilde{\sqcup}$.

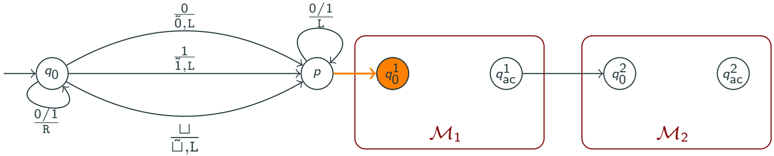
The NTM looks like this:



“Guess” a partition of w into $w = v_1 v_2$ – Illustration

We have new symbols $\tilde{0}, \tilde{1}, \tilde{\sqcup}$.

The NTM looks like this:



Proof of the closure under Kleene star

(Closure under Kleene star) We present a 2-tape NTM $\mathcal{M}$ that recognizes L_1^*.
On input word w:

- "Guess" a partition $v_1 \cdots v_k$ of w, for some $k \geqslant 1$.
- For each $i = 1, \ldots, k$:
  - Copy v_i onto the second tape.
  - Run $\mathcal{M}_1$ on v_i (on the second tape).
  - If $\mathcal{M}_1$ accepts, erase the second tape.
- ACCEPT.

Table of contents

How can we view non-deterministic algorithms?

Non-deterministic algorithms are standard algorithms extended with an instruction of the form:

$$z \quad := \quad 0 \parallel 1;$$

How can we view non-deterministic algorithms?

Non-deterministic algorithms are standard algorithms extended with an instruction of the form:

$$z \quad := \quad 0 \parallel 1;$$

It means "randomly assign variable z with either 0 or 1."

How can we view non-deterministic algorithms?

Non-deterministic algorithms are standard algorithms extended with an instruction of the form:

$$z \quad := \quad 0 \parallel 1;$$

It means "randomly assign variable z with either 0 or 1."

(Def.) A non-deterministic algorithm A "accepts" an input word w, if on every instruction:

$$z \quad := \quad 0 \parallel 1;$$

variable z can be assigned with 0 or 1 such that A will "return true."

Example: The problem SAT

SAT

Input: A propositional formula φ.
Task: Output `True`, if φ has (at least one) satisfying assignment.
Otherwise, output `False`.

Example: The problem SAT

SAT

Input: A propositional formula φ.
Task: Output `True`, if φ has (at least one) satisfying assignment.
Otherwise, output `False`.

(Algo.) On input formula φ:

- Let $x_1, \ldots, x_n$ be the variables in φ.
- For each $i = 1, \ldots, n$ do:
  - $z \; := \; 0 \parallel 1;$
  - If $z == 1$, assign x_i with `True`.
  - If $z == 0$, assign x_i with `False`.
- Check if the formula φ evaluates to true under the assignment.
- If it evaluates to `True`, then ACCEPT.
  If it evaluates to `False`, then REJECT.

Example: The problem Independent-Set

Independent-Set

Input: An undirected graph $G = (V, E)$ and an integer $k \geqslant 1$ (written in binary).
Task: Output `True`, if there is an independent set of k vertices in G.
Otherwise, output `False`.

Example: The problem Independent-Set

Independent-Set

Input:	An undirected graph $G = (V, E)$ and an integer $k \geqslant 1$ (written in binary).
Task:	Output `True`, if there is an independent set of k vertices in G. Otherwise, output `False`.

(Def.) For a graph $G = (V, E)$, a set $S \subseteq V$ is an independent set in G, if every two vertices u, v in S are not adjacent, i.e., $(u, v) \notin E$.

Example: The problem Independent-Set

Independent-Set

Input: An undirected graph $G = (V, E)$ and an integer $k \geqslant 1$ (written in binary).
Task: Output `True`, if there is an independent set of k vertices in G.
Otherwise, output `False`.

(Def.) For a graph $G = (V, E)$, a set $S \subseteq V$ is an independent set in G, if every two vertices u, v in S are not adjacent, i.e., $(u, v) \notin E$.

(Algo.) On input graph $G = (V, E)$ and an integer $k \geqslant 1$:

- $S := \emptyset$.
- For each vertex $v \in V$ do:
  - $z \ := \ 0 \parallel 1$;
  - If $z == 1$, insert v into S.
- Check if the set S is an independent set and $|S| \geqslant k$.
- If it is, ACCEPT.
  If it is not, REJECT.

End of Lesson 9