COL 352 Introduction to Automata and Theory of Computation

Nikhil Balaji

Bharti 420
Indian Institute of Technology, Delhi
nbalaji@cse.iitd.ac.in

April 5, 2023

Lecture 27: Reductions 2

Recap

Undecidability

Recap

Undecidability

$$A_M = \{\langle M, w\rangle \mid w \in L(M)\}$$

$$E_M = \{\langle M\rangle \mid L(M) = \emptyset\}$$

$$EQ_M = \{\langle M_1, M_2\rangle \mid L(M_1) = L(M_2)\}$$

$$REG = \{\langle M\rangle \mid L(M) \text{ is regular}\}$$

Recap

Undecidability

$$A_M = \{\langle M, w\rangle \mid w \in L(M)\}$$

$$E_M = \{\langle M\rangle \mid L(M) = \varnothing\}$$

$$EQ_M = \{\langle M_1, M_2\rangle \mid L(M_1) = L(M_2)\}$$

$$REG = \{\langle M\rangle \mid L(M) \text{ is regular}\}$$

Reduction via computation histories

Definition

For a TM M on input w, $C_1, C_2, \ldots, C_r$ is the accepting computation history (aka accepting run) of M on w if C_1 is the start configuration and C_r is an accepting configuration. Similarly define rejecting computation history (rejecting run). Define graph of configurations.

Reduction via computation histories

Definition

For a TM M on input w, $C_1, C_2, \ldots, C_r$ is the accepting computation history (aka accepting run) of M on w if C_1 is the start configuration and C_r is an accepting configuration. Similarly define rejecting computation history (rejecting run). Define graph of configurations.

- Versatile technique used to prove (un)decidability.
- (Un) decidability = reachability problem on configuration graph.

Reduction via computation histories

Definition

For a TM M on input w, $C_1, C_2, \ldots, C_r$ is the accepting computation history (aka accepting run) of M on w if C_1 is the start configuration and C_r is an accepting configuration. Similarly define rejecting computation history (rejecting run). Define graph of configurations.

- Versatile technique used to prove (un)decidability.
- (Un) decidability = reachability problem on configuration graph.
- Used in the proof of Hilbert's 10th problem (testing integer roots of polynomials) and is a useful technique.

Linearly Bounded Automaton (LBA)

Linearly Bounded Automaton (LBA)

Definition

An LBA is a TM where the tape heard is not allowed to move off the portion of the tape containing the input, i.e. TMs with limited amount of memory($|w|$ bits?)

Linearly Bounded Automaton (LBA)

Definition

An LBA is a TM where the tape heard is not allowed to move off the portion of the tape containing the input, i.e. TMs with limited amount of memory($|w|$ bits?)

- Using a tape alphabet larger than input alphabet, memory available can be $O(n)$ - hence LBA.
- Verifying exercise: The following algorithms from the previous reading exercise ($A_{DFA}, A_{CFG}, E_{DFA}, E_{CFG}$) can be implemented using a LBA.

Linearly Bounded Automaton (LBA)

Definition

An LBA is a TM where the tape heard is not allowed to move off the portion of the tape containing the input, i.e. TMs with limited amount of memory($|w|$ bits?)

- Using a tape alphabet larger than input alphabet, memory available can be $O(n)$ - hence LBA.
- Verifying exercise: The following algorithms from the previous reading exercise ($A_{DFA}, A_{CFG}, E_{DFA}, E_{CFG}$) can be implemented using a LBA.
- Let

$$A_{LBA} = \{(M, w) \mid M \text{ accepts } w\}$$

- Is A_{LBA} undecidable?

Membership for LBA is decidable

Lemma

Given an LBA with q states and $|\Gamma| = g$ there are exactly, qng^n distinct configurations of the LBA.

- Configurations are strings specified by Q, head position and tape contents.

An easy algorithm for A_{LBA}: On input $\langle M, w\rangle$

1. Simulate M on w for qng^n steps or until it halts.
2. If M halts accept (with accept/reject based on M) else reject

What about E_{LBA}?

$$E_{LBA} = \{\langle M\rangle \mid L(M) = \varnothing\}$$

Emptiness for LBA is undecidable!

Emptiness for LBA is undecidable!

- Reduce A_{TM} to E_{LBA}.

Emptiness for LBA is undecidable!

- Reduce A_{TM} to E_{LBA}.
- I.e., Given $\langle M, w \rangle$ can we build a LBA B such that $L(B)$ is empty if and only if M accepts w?

Emptiness for LBA is undecidable!

- Reduce A_{TM} to E_{LBA}.
- I.e., Given $\langle M, w\rangle$ can we build a LBA B such that $L(B)$ is empty if and only if M accepts w?
- Idea: Language that B recognizes should be language of accepting runs (computation histories) of M on w.

Emptiness for LBA is undecidable!

- Reduce A_{TM} to E_{LBA}.
- I.e., Given $\langle M, w \rangle$ can we build a LBA B such that $L(B)$ is empty if and only if M accepts w?
- Idea: Language that B recognizes should be language of accepting runs (computation histories) of M on w.
- Assume computation histories are specified as $\#C_1\#C_2\#\ldots C_r\#$.

Emptiness for LBA is undecidable!

- Reduce A_{TM} to E_{LBA}.
- I.e., Given $\langle M, w \rangle$ can we build a LBA B such that $L(B)$ is empty if and only if M accepts w?
- Idea: Language that B recognizes should be language of accepting runs (computation histories) of M on w.
- Assume computation histories are specified as $\#C_1\#C_2\#\ldots C_r\#$.
- The LBA has to check if C_i's satisfy

Emptiness for LBA is undecidable!

- Reduce A_{TM} to E_{LBA}.
- I.e., Given $\langle M, w\rangle$ can we build a LBA B such that $L(B)$ is empty if and only if M accepts w?
- Idea: Language that B recognizes should be language of accepting runs (computation histories) of M on w.
- Assume computation histories are specified as $\#C_1\#C_2\#\ldots C_r\#$.
- The LBA has to check if C_i's satisfy
  1. C_1 is the start configuration.
  2. C_{i+1} legally follows from C_i.
  3. C_r is an accepting configuration.

Emptiness for LBA is undecidable!

- Reduce A_{TM} to E_{LBA}.
- I.e., Given $\langle M, w\rangle$ can we build a LBA B such that $L(B)$ is empty if and only if M accepts w?
- Idea: Language that B recognizes should be language of accepting runs (computation histories) of M on w.
- Assume computation histories are specified as $\#C_1\#C_2\#\ldots C_r\#$.
- The LBA has to check if C_i's satisfy
  1. C_1 is the start configuration.
  2. C_{i+1} legally follows from C_i.
  3. C_r is an accepting configuration.
- Remember B has M's code and w hard-coded (as input!) (Condition 1 is easy)

Emptiness for LBA is undecidable!

- Reduce A_{TM} to E_{LBA}.
- I.e., Given $\langle M, w\rangle$ can we build a LBA B such that $L(B)$ is empty if and only if M accepts w?
- Idea: Language that B recognizes should be language of accepting runs (computation histories) of M on w.
- Assume computation histories are specified as $\#C_1\#C_2\#\ldots C_r\#$.
- The LBA has to check if C_i's satisfy
  1. C_1 is the start configuration.
  2. C_{i+1} legally follows from C_i.
  3. C_r is an accepting configuration.
- Remember B has M's code and w hard-coded (as input!) (Condition 1 is easy)
- C_{i+1} cannot be too different from C_i: except the spots near the head of the machine (easy to verify from the code of M - mark with dots, zig-zag).

Emptiness for LBA is undecidable!

- Assume there exists an LBA B that correctly identifies accepting runs of TM M on input w.
- Let R be algorithm (assumed) to exist for deciding E_{LBA}

Emptiness for LBA is undecidable!

- Assume there exists an LBA B that correctly identifies accepting runs of TM M on input w.
- Let R be algorithm (assumed) to exist for deciding E_{LBA}

Here is the algorithm

On input $\langle M, w\rangle$,

1. Construct LBA B
2. Run R on $\langle B\rangle$
3. If R rejects, accept, else if R accepts, reject.

Universality of CFG

Lemma

$ALL_{CFG} = \{\langle M \rangle \mid M$ *is a PDA and* $L(M) = \Sigma^*\}$ *is undecidable.*

Universality of CFG

Lemma

$ALL_{CFG} = \{\langle M \rangle \mid M$ *is a PDA and* $L(M) = \Sigma^*\}$ *is undecidable.*

Proof Strategy

Universality of CFG

Lemma

$ALL_{CFG} = \{\langle M \rangle \mid M$ *is a PDA and* $L(M) = \Sigma^*\}$ *is undecidable.*

Proof Strategy

For a TM M and input w we create a PDA $N_{M,w}$ such that

Universality of CFG

Lemma

$ALL_{CFG} = \{\langle M \rangle \mid M$ *is a PDA and* $L(M) = \Sigma^*\}$ *is undecidable.*

Proof Strategy

For a TM M and input w we create a PDA $N_{M,w}$ such that

$N_{M,w}$ accepts all strings

Universality of CFG

Lemma

$ALL_{CFG} = \{\langle M \rangle \mid M$ is a PDA and $L(M) = \Sigma^*\}$ is undecidable.

Proof Strategy

For a TM M and input w we create a PDA $N_{M,w}$ such that

$N_{M,w}$ accepts all strings (i.e. accepts Σ^*)

Universality of CFG

Lemma

$ALL_{CFG} = \{\langle M \rangle \mid M$ is a PDA and $L(M) = \Sigma^*\}$ is undecidable.

Proof Strategy

For a TM M and input w we create a PDA $N_{M,w}$ such that

$N_{M,w}$ accepts all strings (i.e. accepts Σ^*) if M accepts w

Universality of CFG

Lemma

$ALL_{CFG} = \{\langle M \rangle \mid M$ is a PDA and $L(M) = \Sigma^*\}$ is undecidable.

Proof Strategy

For a TM M and input w we create a PDA $N_{M,w}$ such that

$N_{M,w}$ accepts all strings (i.e. accepts Σ^*) if M accepts w, and

$N_{M,w}$ rejects at least one string if M does not accept w.

Universality of CFG

Lemma

$ALL_{CFG} = \{\langle M\rangle \mid M$ is a PDA and $L(M) = \Sigma^*\}$ is undecidable.

Proof Strategy

For a TM M and input w we create a PDA $N_{M,w}$ such that

$N_{M,w}$ accepts all strings (i.e. accepts Σ^*) if M accepts w, and

$N_{M,w}$ rejects at least one string if M does not accept w.

Formally,

Universality of CFG

Lemma

$ALL_{CFG} = \{\langle M \rangle \mid M$ *is a PDA and* $L(M) = \Sigma^*\}$ *is undecidable.*

Proof Strategy

For a TM M and input w we create a PDA $N_{M,w}$ such that

$N_{M,w}$ accepts all strings (i.e. accepts Σ^*) if M accepts w, and

$N_{M,w}$ rejects at least one string if M does not accept w.

Formally,

Input $(M, w) \quad \longrightarrow \quad N_{M,w}$

Universality of CFG

Lemma

$ALL_{CFG} = \{\langle M \rangle \mid M$ *is a PDA and* $L(M) = \Sigma^*\}$ *is undecidable.*

Proof Strategy

For a TM M and input w we create a PDA $N_{M,w}$ such that

$N_{M,w}$ accepts all strings (i.e. accepts Σ^*) if M accepts w, and

$N_{M,w}$ rejects at least one string if M does not accept w.

Formally,

Input $(M, w) \quad \longrightarrow \quad N_{M,w}$

if $w \in L(M) \quad \longrightarrow \quad \exists x \in \Sigma^*,$ s.t. $x \notin L(N_{M,w})$

Universality of CFG

Lemma

$ALL_{CFG} = \{\langle M \rangle \mid M$ *is a PDA and* $L(M) = \Sigma^*\}$ *is undecidable.*

Proof Strategy

For a TM M and input w we create a PDA $N_{M,w}$ such that

$N_{M,w}$ accepts all strings (i.e. accepts Σ^*) if M accepts w, and

$N_{M,w}$ rejects at least one string if M does not accept w.

Formally,

Input $(M, w) \quad \longrightarrow \quad N_{M,w}$

if $w \in L(M) \quad \longrightarrow \quad \exists x \in \Sigma^*,$ s.t. $x \notin L(N_{M,w})$

if $w \notin L(M) \quad \longrightarrow \quad L(N_{M,w}) = \Sigma^*$

Universality of CFG

Lemma

$ALL_{CFG} = \{\langle M \rangle \mid M$ is a PDA and $L(M) = \Sigma^*\}$ is undecidable.

Proof Strategy

For a TM M and input w we create a PDA $N_{M,w}$ such that

$N_{M,w}$ accepts all strings (i.e. accepts Σ^*) if M accepts w, and

$N_{M,w}$ rejects at least one string if M does not accept w.

Formally,

Input $(M, w) \quad \longrightarrow \quad N_{M,w}$

if $w \in L(M) \quad \longrightarrow \quad \exists x \in \Sigma^*,$ s.t. $x \notin L(N_{M,w})$

if $w \notin L(M) \quad \longrightarrow \quad L(N_{M,w}) = \Sigma^*$

Filling in the details

The following two details need to be addressed.

Q_1 How should we design $N_{M,w}$?

Filling in the details

The following two details need to be addressed.

Q_1 How should we design $N_{M,w}$?

Q_2 If such an $N_{M,w}$ is designed then why have we proved that ALL_{CFL} is undecidable?

Details for Q_2

Q_2 If such an $N_{M,w}$ is designed then why have we proved that ALL_{CFL} is undecidable?

Input $(M, w) \quad \longrightarrow \quad N_{M,w}$

Details for Q_2

Q_2 If such an $N_{M,w}$ is designed then why have we proved that ALL_{CFL} is undecidable?

Input (M, w)	$\longrightarrow$	$N_{M,w}$
if $w \in L(M)$	$\longrightarrow$	$\exists x \in \Sigma^*,$ s.t. $x \notin L(N_{M,w})$

Details for Q_2

Q_2 If such an $N_{M,w}$ is designed then why have we proved that ALL_{CFL} is undecidable?

$$\begin{array}{lcl} \text{Input } (M, w) & \longrightarrow & N_{M,w} \\ \text{if } w \in L(M) & \longrightarrow & \exists x \in \Sigma^*, \text{ s.t. } x \notin L(N_{M,w}) \\ \text{if } w \notin L(M) & \longrightarrow & L(N_{M,w}) = \Sigma^* \end{array}$$

Details for Q_2

Q_2 If such an $N_{M,w}$ is designed then why have we proved that ALL_{CFL} is undecidable?

$$\begin{array}{lcl} \text{Input } (M, w) & \longrightarrow & N_{M,w} \\ \text{if } w \in L(M) & \longrightarrow & \exists x \in \Sigma^*, \text{ s.t. } x \notin L(N_{M,w}) \\ \text{if } w \notin L(M) & \longrightarrow & L(N_{M,w}) = \Sigma^* \end{array}$$

Assume that ALL_{CFL} is decidable.

Details for Q_2

Q_2 If such an $N_{M,w}$ is designed then why have we proved that ALL_{CFL} is undecidable?

$$\begin{array}{lcl} \text{Input } (M,w) & \longrightarrow & N_{M,w} \\ \text{if } w \in L(M) & \longrightarrow & \exists x \in \Sigma^*, \text{ s.t. } x \notin L(N_{M,w}) \\ \text{if } w \notin L(M) & \longrightarrow & L(N_{M,w}) = \Sigma^* \end{array}$$

Assume that ALL_{CFL} is decidable.
C be the TM deciding it.

Details for Q_2

Q_2 If such an $N_{M,w}$ is designed then why have we proved that ALL_{CFL} is undecidable?

Design A as follows:

For an M, w pair, create $N_{M,w}$.

Input (M, w) $\longrightarrow$ $N_{M,w}$

if $w \in L(M)$ $\longrightarrow$ $\exists x \in \Sigma^*,$ s.t. $x \notin L(N_{M,w})$

if $w \notin L(M)$ $\longrightarrow$ $L(N_{M,w}) = \Sigma^*$

Assume that ALL_{CFL} is decidable.
C be the TM deciding it.

Details for Q_2

Q_2 If such an $N_{M,w}$ is designed then why have we proved that ALL_{CFL} is undecidable?

Input (M, w)	$\longrightarrow$	$N_{M,w}$
if $w \in L(M)$	$\longrightarrow$	$\exists x \in \Sigma^*,$ s.t. $x \notin L(N_{M,w})$
if $w \notin L(M)$	$\longrightarrow$	$L(N_{M,w}) = \Sigma^*$

Assume that ALL_{CFL} is decidable.
C be the TM deciding it.

Design A as follows:

For an M, w pair, create $N_{M,w}$.

Feed $\langle N_{M,w} \rangle$ to C.

Details for Q_2

Q_2 If such an $N_{M,w}$ is designed then why have we proved that ALL_{CFL} is undecidable?

Input $(M, w) \longrightarrow N_{M,w}$

if $w \in L(M) \longrightarrow \exists x \in \Sigma^*,$ s.t. $x \notin L(N_{M,w})$

if $w \notin L(M) \longrightarrow L(N_{M,w}) = \Sigma^*$

Assume that ALL_{CFL} is decidable.
C be the TM deciding it.

Design A as follows:

For an M, w pair, create $N_{M,w}$.

Feed $\langle N_{M,w} \rangle$ to C.

If C accepts then reject;

Details for Q_2

Q_2 If such an $N_{M,w}$ is designed then why have we proved that ALL_{CFL} is undecidable?

Input (M, w)	$\longrightarrow$	$N_{M,w}$
if $w \in L(M)$	$\longrightarrow$	$\exists x \in \Sigma^*,$ s.t. $x \notin L(N_{M,w})$
if $w \notin L(M)$	$\longrightarrow$	$L(N_{M,w}) = \Sigma^*$

Assume that ALL_{CFL} is decidable.
C be the TM deciding it.

Design A as follows:

For an M, w pair, create $N_{M,w}$.

Feed $\langle N_{M,w} \rangle$ to C.

If C accepts then reject;

else accept.

Details for Q_2

Q_2 If such an $N_{M,w}$ is designed then why have we proved that ALL_{CFL} is undecidable?

$$\begin{array}{lcl} \text{Input } (M,w) & \longrightarrow & N_{M,w} \\ \text{if } w \in L(M) & \longrightarrow & \exists x \in \Sigma^*, \text{ s.t. } x \notin L(N_{M,w}) \\ \text{if } w \notin L(M) & \longrightarrow & L(N_{M,w}) = \Sigma^* \end{array}$$

Assume that ALL_{CFL} is decidable.
C be the TM deciding it.

Design A as follows:

For an M, w pair, create $N_{M,w}$.

Feed $\langle N_{M,w} \rangle$ to C.

If C accepts then reject;

else accept.

Details for Q_1: reduction via computation history

Q_1 How should we design $N_{M,w}$?

Details for Q_1: reduction via computation history

Q_1 How should we design $N_{M,w}$?

Main idea

Use computational history of M on w.

Details for Q_1: reduction via computation history

Q_1 How should we design $N_{M,w}$?

Main idea

Use computational history of M on w.

Accepting computation history is a sequece of configutations: $C_1, C_2, \ldots, C_\ell$ such that

Details for Q_1: reduction via computation history

Q_1 How should we design $N_{M,w}$?

Main idea

Use computational history of M on w.

Accepting computation history is a sequece of configutations: $C_1, C_2, \ldots, C_\ell$ such that

- C_1 is a start configutation.
- C_ℓ is an accepting configuration.
- for each $1 \le i \le \ell$, C_i yields C_{i+1}.

Rejecting computation history is a sequece of configutations: $C_1, C_2, \ldots, C_\ell$ such that

Details for Q_1: reduction via computation history

Q_1 How should we design $N_{M,w}$?

Main idea

Use computational history of M on w.

Accepting computation history is a sequece of configutations: $C_1, C_2, \dots, C_\ell$ such that

- C_1 is a start configutation.
- C_ℓ is an accepting configuration.
- for each $1 \le i \le \ell$, C_i yields C_{i+1}.

Rejecting computation history is a sequece of configutations: $C_1, C_2, \dots, C_\ell$ such that

- C_1 is a start configutation.
- C_ℓ is a rejecting configuration.
- for each $1 \le i \le \ell$, C_i yields C_{i+1}.

Universality of CFLs is undecidable!

- Interpret input x to $N_{M,w}$ as a computational history of M on w.

Universality of CFLs is undecidable!

- Interpret input x to $N_{M,w}$ as a computational history of M on w.
- Design $N_{M,w}$ s.t. it accepts x if any of the following conditions holds:

Universality of CFLs is undecidable!

- Interpret input x to $N_{M,w}$ as a computational history of M on w.
- Design $N_{M,w}$ s.t. it accepts x if any of the following conditions holds:
  - x does not have the pattern of a computational history of x

Universality of CFLs is undecidable!

- Interpret input x to $N_{M,w}$ as a computational history of M on w.
- Design $N_{M,w}$ s.t. it accepts x if any of the following conditions holds:
  - x does not have the pattern of a computational history of x OR
  - x is a computational history, but C_1 is not a start configuration

Universality of CFLs is undecidable!

- Interpret input x to $N_{M,w}$ as a computational history of M on w.
- Design $N_{M,w}$ s.t. it accepts x if any of the following conditions holds:
  - x does not have the pattern of a computational history of x OR
  - x is a computational history, but C_1 is not a start configuration OR
  - x is a computational history, C_1 is a start configuration, but C_ℓ is not an accepting configutation

Universality of CFLs is undecidable!

- Interpret input x to $N_{M,w}$ as a computational history of M on w.
- Design $N_{M,w}$ s.t. it accepts x if any of the following conditions holds:
  - x does not have the pattern of a computational history of x OR
  - x is a computational history, but C_1 is not a start configuration OR
  - x is a computational history, C_1 is a start configuration, but C_ℓ is not an accepting configutation OR
  - x is a computational history, C_1 is a start configuration, C_ℓ is an accepting configutation, but there exists an i s.t. $1 \leq i \leq \ell - 1$ and C_i does not yield C_{i+1}.

Universality of CFLs is undecidable!

- Interpret input x to $N_{M,w}$ as a computational history of M on w.
- Design $N_{M,w}$ s.t. it accepts x if any of the following conditions holds:
  - x does not have the pattern of a computational history of x OR
  - x is a computational history, but C_1 is not a start configuration OR
  - x is a computational history, C_1 is a start configuration, but C_ℓ is not an accepting configutation OR
  - x is a computational history, C_1 is a start configuration, C_ℓ is an accepting configutation, but there exists an i s.t. $1 \le i \le \ell - 1$ and C_i does not yield C_{i+1}.
- If M accepts w, let $\tilde{x}$ be a accepting computation history of M on w.

  $N_{M,w}$ will reject $\tilde{x}$

Universality of CFLs is undecidable!

- Interpret input x to $N_{M,w}$ as a computational history of M on w.
- Design $N_{M,w}$ s.t. it accepts x if any of the following conditions holds:
  - x does not have the pattern of a computational history of x OR
  - x is a computational history, but C_1 is not a start configuration OR
  - x is a computational history, C_1 is a start configuration, but C_ℓ is not an accepting configutation OR
  - x is a computational history, C_1 is a start configuration, C_ℓ is an accepting configutation, but there exists an i s.t. $1 \leq i \leq \ell - 1$ and C_i does not yield C_{i+1}.
- If M accepts w, let $\tilde{x}$ be a accepting computation history of M on w.

  $N_{M,w}$ will reject $\tilde{x}$, i.e. $\tilde{x} \notin L(N_{M,w})$.

Universality of CFLs is undecidable!

- Interpret input x to $N_{M,w}$ as a computational history of M on w.
- Design $N_{M,w}$ s.t. it accepts x if any of the following conditions holds:
  - x does not have the pattern of a computational history of x OR
  - x is a computational history, but C_1 is not a start configuration OR
  - x is a computational history, C_1 is a start configuration, but C_ℓ is not an accepting configutation OR
  - x is a computational history, C_1 is a start configuration, C_ℓ is an accepting configutation, but there exists an i s.t. $1 \leq i \leq \ell - 1$ and C_i does not yield C_{i+1}.
- If M accepts w, let $\tilde{x}$ be a accepting computation history of M on w.

  $N_{M,w}$ will reject $\tilde{x}$, i.e. $\tilde{x} \notin L(N_{M,w})$.
- If M does not accept w, then

  no matter what x is, $N_{M,w}$ will accept x

Universality of CFLs is undecidable!

- Interpret input x to $N_{M,w}$ as a computational history of M on w.
- Design $N_{M,w}$ s.t. it accepts x if any of the following conditions holds:
  - x does not have the pattern of a computational history of x OR
  - x is a computational history, but C_1 is not a start configuration OR
  - x is a computational history, C_1 is a start configuration, but C_ℓ is not an accepting configutation OR
  - x is a computational history, C_1 is a start configuration, C_ℓ is an accepting configutation, but there exists an i s.t. $1 \leq i \leq \ell - 1$ and C_i does not yield C_{i+1}.
- If M accepts w, let $\tilde{x}$ be a accepting computation history of M on w.

  $N_{M,w}$ will reject $\tilde{x}$, i.e. $\tilde{x} \notin L(N_{M,w})$.
- If M does not accept w, then

  no matter what x is, $N_{M,w}$ will accept x, i.e. $L(N_{M,w}) = \Sigma^*$.