بسم الله الرحمن الرحیم

نظریه علوم کامپیوتر

نظریه علوم کامپیوتر - بهار ۱۴۰۰-۱۴۰۱ - جلسه پنجم: محاسبه‌پذیری و محاسبه‌ناپذیری

Theory of computation - 002 - S05 - non-computability

TMs and Encodings – review

A TM has 3 possible outcomes for each input w:

1. *Accept* w (enter $q$acc)
2. *Reject* w by halting (enter $q$rej)
3. *Reject* w by looping (running forever)

TMs and Encodings – review

A TM has 3 possible outcomes for each input w:

1. *Accept* w (enter $q$acc)
2. *Reject* w by halting (enter $q$rej)
3. *Reject* w by looping (running forever)

A is T-recognizable if $A = L(M)$ for some TM M.

A is T-decidable if $A = L(M)$ for some TM decider M.

halts on all inputs

TMs and Encodings – review

A TM has 3 possible outcomes for each input w:

1. *Accept* w (enter $q$acc)
2. *Reject* w by halting (enter $q$rej)
3. *Reject* w by looping (running forever)

A is T-recognizable if $A = L(M)$ for some TM M.

A is T-decidable if $A = L(M)$ for some TM decider M.

halts on all inputs

$\langle O_1,\ O_2,\ \ldots,\ O_k \rangle$ encodes objects $O_1,\ O_2,\ \ldots,\ O_k$ as a single string.

TMs and Encodings – review

A TM has 3 possible outcomes for each input w:

1. *Accept* w (enter $q$acc)
2. *Reject* w by halting (enter $q$rej)
3. *Reject* w by looping (running forever)

A is T-recognizable if $A = L(M)$ for some TM M.

A is T-decidable if $A = L(M)$ for some TM decider M.

halts on all inputs

$\langle O_1,\ O_2,\ \ldots,\ O_k \rangle$ encodes objects $O_1,\ O_2,\ \ldots,\ O_k$ as a single string.

Notation for writing a TM M is

$M =$ "On input w

[English description of the algorithm]"

TMs and Encodings – review

A TM has 3 possible outcomes for each input w:

1. *Accept* w (enter $q$acc)
2. *Reject* w by halting (enter $q$rej)
3. *Reject* w by looping (running forever)

A is T-recognizable if $A = L(M)$ for some TM M.

A is T-decidable if $A = L(M)$ for some TM decider M.

halts on all inputs

$\langle O_1,\ O_2,\ \ldots,\ O_k \rangle$ encodes objects $O_1,\ O_2,\ \ldots,\ O_k$ as a single string.

Notation for writing a TM M is

$M =$ "On input w

[English description of the algorithm]"

TMs and Encodings – review

A TM has 3 possible outcomes for each input w:

1. *Accept* w (enter $q$acc)
2. *Reject* w by halting (enter $q$rej)
3. *Reject* w by looping (running forever)

A is T-recognizable if $A = L(M)$ for some TM M.

A is T-decidable if $A = L(M)$ for some TM decider M.

halts on all inputs

$\langle O_1,\ O_2,\ \ldots,\ O_k \rangle$ encodes objects $O_1,\ O_2,\ \ldots,\ O_k$ as a single string.

Notation for writing a TM M is

$M =$ "On input w

[English description of the algorithm]"

Acceptance Problem for DFAs

Let ADFA $= \{\langle B, w\rangle \mid B$ is a DFA and B accepts $w\}$

Acceptance Problem for DFAs

Let ADFA $= \{\langle B, w\rangle \mid B$ is a DFA and B accepts $w\}$

Theorem: ADFA is decidable

Acceptance Problem for DFAs

Let $A\text{DFA} = \left\{ \langle B, w \rangle \,\middle|\, B \text{ is a DFA and } B \text{ accepts } w \right\}$

Theorem: $A\text{DFA}$ is decidable

Proof: Give TM $D\text{A−DFA}$ that decides $A\text{DFA}$.

Acceptance Problem for DFAs

Let $A_{\text{DFA}} = \{\langle B, w\rangle \mid B \text{ is a DFA and } B \text{ accepts } w\}$

Theorem: A_{DFA} is decidable

Proof: Give TM $D_{\text{A-DFA}}$ that decides A_{DFA} .

$D_{\text{A-DFA}} =$ "On input s

Acceptance Problem for DFAs

Let $A_{\text{DFA}} = \{\langle B, w \rangle \mid B \text{ is a DFA and } B \text{ accepts } w\}$

Theorem: A_{DFA} is decidable

Proof: Give TM $D_{\text{A-DFA}}$ that decides A_{DFA}.

$D_{\text{A-DFA}} =$ "On input s

1. Check that s has the form $\langle B, w \rangle$ where B is a DFA and w is a string; *reject* if not.

Acceptance Problem for DFAs

Let $A_{\text{DFA}} = \{\langle B, w\rangle \mid B$ is a DFA and B accepts $w\}$

Theorem: A_{DFA} is decidable

Proof: Give TM $D_{\text{A-DFA}}$ that decides A_{DFA}.

$D_{\text{A-DFA}} =$ "On input s

1. Check that s has the form $\langle B, w\rangle$ where B is a DFA and w is a string; *reject* if not.
2. Simulate the computation of B on w.

Acceptance Problem for DFAs

Let $A_{\text{DFA}} = \{\langle B, w\rangle \mid B$ is a DFA and B accepts $w\}$

Theorem: A_{DFA} is decidable

Proof: Give TM $D_{\text{A-DFA}}$ that decides A_{DFA} .

$D_{\text{A-DFA}} =$ "On input s

1. Check that s has the form $\langle B, w\rangle$ where B is a DFA and w is a string; *reject* if not.
2. Simulate the computation of B on w.
3. If B ends in an accept state then *accept*. If not then *reject*."

Acceptance Problem for DFAs

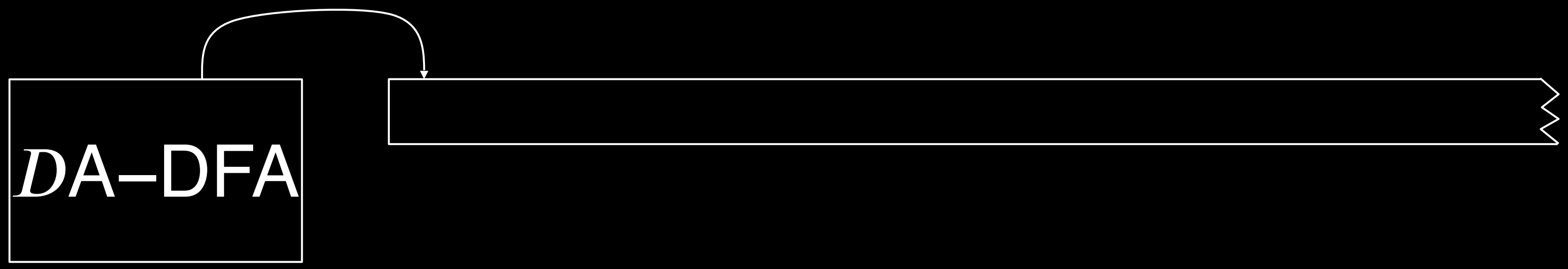
Let $A_{\text{DFA}} = \left\{ \langle B, w \rangle \;\middle|\; B \text{ is a DFA and } B \text{ accepts } w \right\}$

Theorem: A_{DFA} is decidable

Proof: Give TM $D_{\text{A-DFA}}$ that decides A_{DFA} .

$D_{\text{A-DFA}} =$ "On input s

1. Check that s has the form $\langle B, w \rangle$ where B is a DFA and w is a string; *reject* if not.
2. Simulate the computation of B on w.
3. If B ends in an accept state then *accept*. If not then *reject*."



Acceptance Problem for DFAs

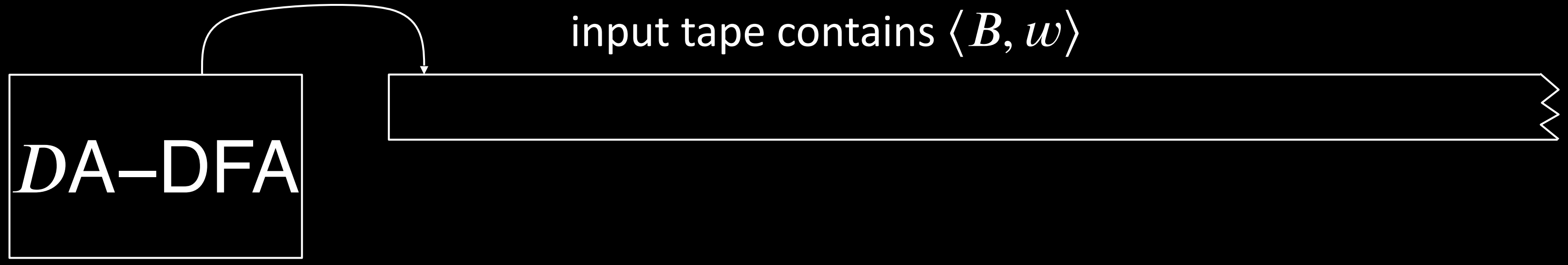
Let $A_{\text{DFA}} = \{\langle B, w\rangle \mid B$ is a DFA and B accepts $w\}$

Theorem: A_{DFA} is decidable

Proof: Give TM $D_{\text{A-DFA}}$ that decides A_{DFA} .

$D_{\text{A-DFA}} =$ "On input s

1. Check that s has the form $\langle B, w\rangle$ where B is a DFA and w is a string; *reject* if not.
2. Simulate the computation of B on w.
3. If B ends in an accept state then *accept*. If not then *reject*."



Acceptance Problem for DFAs

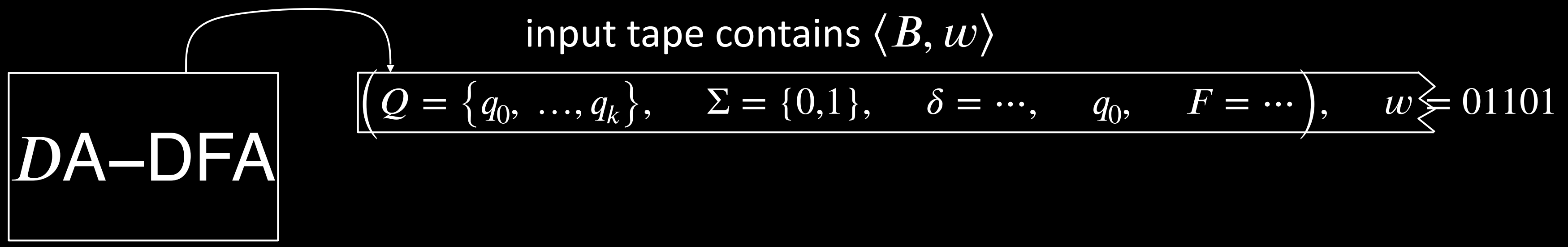
Let $A\text{DFA} = \{\langle B, w\rangle \mid B$ is a DFA and B accepts $w\}$

Theorem: ADFA is decidable

Proof: Give TM DA−DFA that decides ADFA .

DA−DFA = "On input s

1. Check that s has the form $\langle B, w\rangle$ where B is a DFA and w is a string; *reject* if not.
2. Simulate the computation of B on w.
3. If B ends in an accept state then *accept*. If not then *reject*."



Acceptance Problem for DFAs

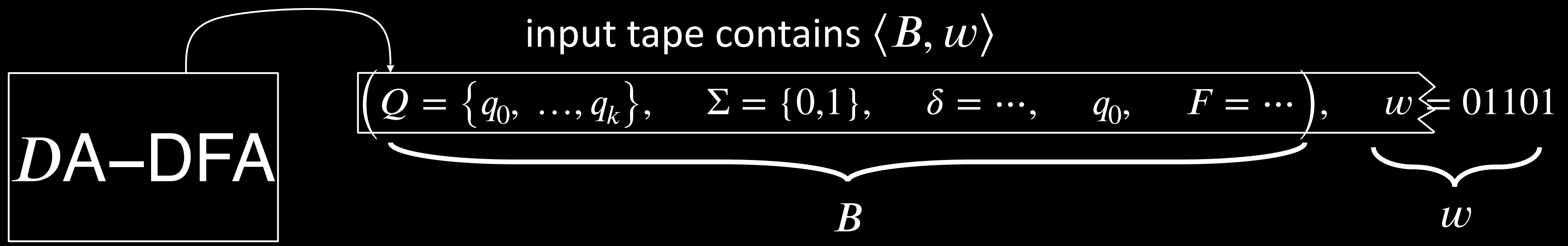
Let $A_{\text{DFA}} = \{\langle B, w\rangle \mid B \text{ is a DFA and } B \text{ accepts } w\}$

Theorem: A_{DFA} is decidable

Proof: Give TM $D_{A\text{-DFA}}$ that decides A_{DFA} .

$D_{A\text{-DFA}} =$ "On input s

1. Check that s has the form $\langle B, w\rangle$ where B is a DFA and w is a string; *reject* if not.
2. Simulate the computation of B on w.
3. If B ends in an accept state then *accept*. If not then *reject*."



Acceptance Problem for DFAs

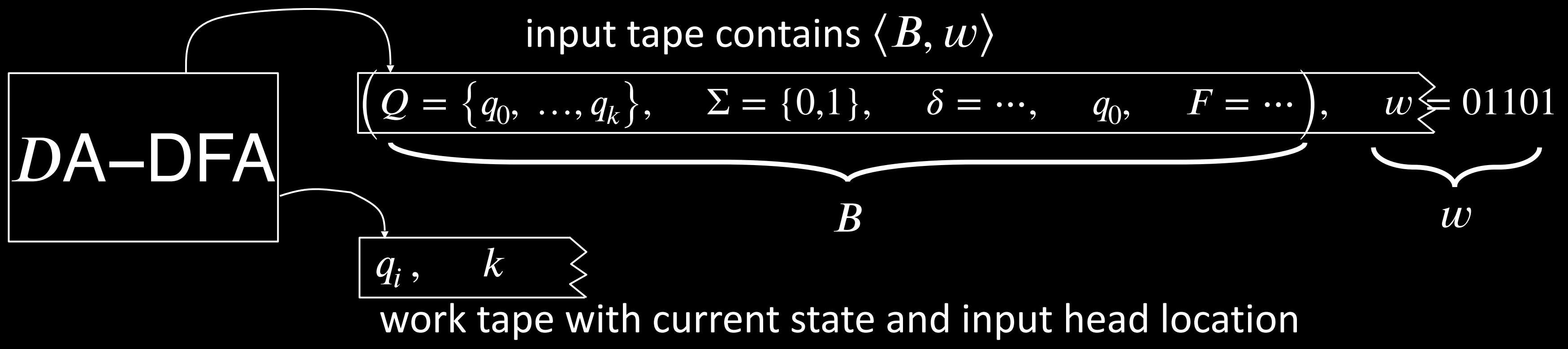
Let $A\text{DFA} = \{\langle B, w\rangle \mid B$ is a DFA and B accepts $w\}$

Theorem: $A\text{DFA}$ is decidable

Proof: Give TM $D_{A\text{-DFA}}$ that decides $A\text{DFA}$.

$D_{A\text{-DFA}} =$ "On input s

1. Check that s has the form $\langle B, w\rangle$ where B is a DFA and w is a string; *reject* if not.
2. Simulate the computation of B on w.
3. If B ends in an accept state then *accept*. If not then *reject*."



Acceptance Problem for DFAs

Let $A_{\text{DFA}} = \{\langle B, w\rangle \mid B$ is a DFA and B accepts $w\}$

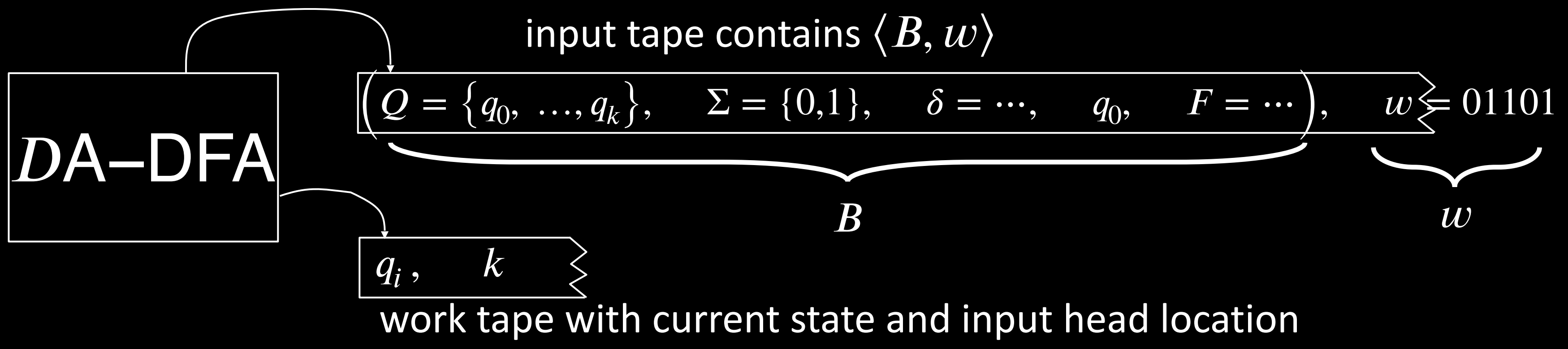
Theorem: A_{DFA} is decidable

Proof: Give TM $D_{\text{A-DFA}}$ that decides A_{DFA} .

$D_{\text{A-DFA}} =$ "On input s

1. Check that s has the form $\langle B, w\rangle$ where B is a DFA and w is a string; *reject* if not.
2. Simulate the computation of B on w.
3. If B ends in an accept state then *accept*. If not then *reject*."

Shorthand:
On input $\langle B, w\rangle$



Acceptance Problem for DFAs

Let $A\text{DFA} = \{\langle B, w\rangle \mid B$ is a DFA and B accepts $w\}$

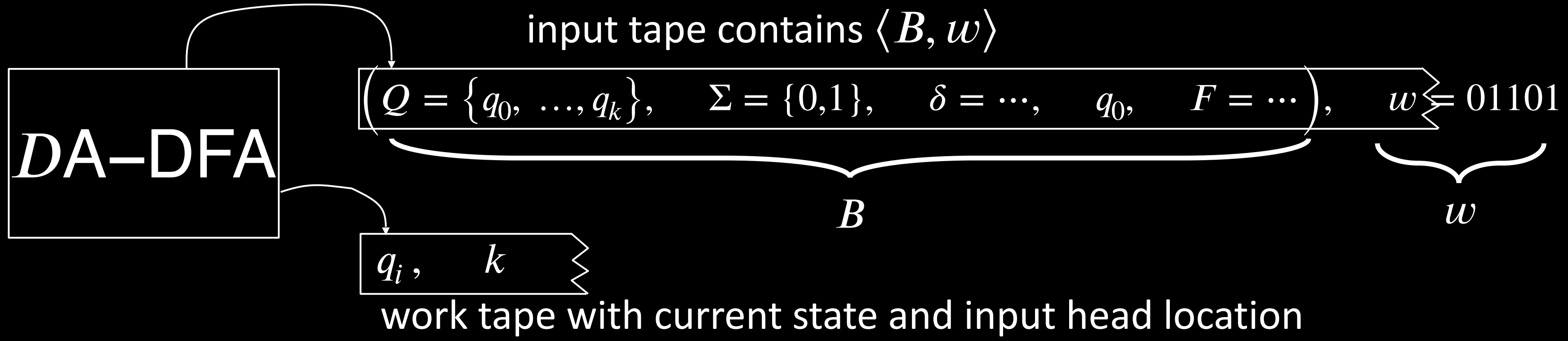
Theorem: ADFA is decidable

Proof: Give TM DA–DFA that decides ADFA .

DA–DFA = "On input s

1. Check that s has the form $\langle B, w\rangle$ where B is a DFA and w is a string; *reject* if not.
2. Simulate the computation of B on w.
3. If B ends in an accept state then *accept*. If not then *reject*."

Shorthand:
On input $\langle B, w\rangle$



Acceptance Problem for NFAs

Let ANFA $= \left\{ \langle B, w \rangle \,\middle|\, B \text{ is a NFA and } B \text{ accepts } w \right\}$

Acceptance Problem for NFAs

Let ANFA $= \{\langle B, w\rangle \mid B$ is a NFA and B accepts $w\}$

Theorem: ANFA is decidable

Acceptance Problem for NFAs

Let $A\text{NFA} = \left\{ \langle B, w \rangle \,\middle|\, B \text{ is a NFA and } B \text{ accepts } w \right\}$

Theorem: ANFA is decidable

Proof: Give TM DA−NFA that decides ANFA .

Acceptance Problem for NFAs

Let $A\text{NFA} = \{\langle B, w\rangle \mid B \text{ is a NFA and } B \text{ accepts } w\}$

Theorem: ANFA is decidable

Proof: Give TM DA–NFA that decides ANFA .

$D\text{A–NFA} =$ “On input $\langle B, w\rangle$

Acceptance Problem for NFAs

Let $A_{\text{NFA}} = \{\langle B, w\rangle \mid B \text{ is a NFA and } B \text{ accepts } w\}$

Theorem: A_{NFA} is decidable

Proof: Give TM $D_{\text{A-NFA}}$ that decides A_{NFA} .

$D_{\text{A-NFA}} =$ "On input $\langle B, w\rangle$

1. Convert NFA B to equivalent DFA B'.

Acceptance Problem for NFAs

Let $A\text{NFA} = \left\{ \langle B, w \rangle \middle| \ B \text{ is a NFA and } B \text{ accepts } w \right\}$

Theorem: ANFA is decidable

Proof: Give TM DA−NFA that decides ANFA .

$D\text{A–NFA} =$ "On input $\langle B, w \rangle$

1. Convert NFA B to equivalent DFA B'.
2. Run TM DA−DFA on input $\langle B', w \rangle$. [Recall that DA−DFA decides ADFA]

Acceptance Problem for NFAs

Let $A_{\text{NFA}} = \{\langle B, w\rangle \mid B \text{ is a NFA and } B \text{ accepts } w\}$

Theorem: A_{NFA} is decidable

Proof: Give TM $D_{\text{A-NFA}}$ that decides A_{NFA}.

$D_{\text{A-NFA}} =$ "On input $\langle B, w\rangle$

1. Convert NFA B to equivalent DFA B'.
2. Run TM $D_{\text{A-DFA}}$ on input $\langle B', w\rangle$. [Recall that $D_{\text{A-DFA}}$ decides A_{DFA}]
3. *Accept* if $D_{\text{A-DFA}}$ accepts.
   Reject if not."

Acceptance Problem for NFAs

Let $A_{\text{NFA}} = \{\langle B, w\rangle \mid B$ is a NFA and B accepts $w\}$

Theorem: A_{NFA} is decidable

Proof: Give TM $D_{\text{A-NFA}}$ that decides A_{NFA} .

$D_{\text{A-NFA}} =$ "On input $\langle B, w\rangle$

1. Convert NFA B to equivalent DFA B'.
2. Run TM $D_{\text{A-DFA}}$ on input $\langle B', w\rangle$. [Recall that $D_{\text{A-DFA}}$ decides A_{DFA}]
3. *Accept* if $D_{\text{A-DFA}}$ accepts.
   Reject if not."

New element: Use conversion construction and previously constructed TM as a subroutine.

Acceptance Problem for NFAs

Let $A_{\text{NFA}} = \{\langle B, w\rangle \mid B$ is a NFA and B accepts $w\}$

Theorem: A_{NFA} is decidable

Proof: Give TM $D_{\text{A-NFA}}$ that decides A_{NFA} .

$D_{\text{A-NFA}} =$ "On input $\langle B, w\rangle$

1. Convert NFA B to equivalent DFA B'.
2. Run TM $D_{\text{A-DFA}}$ on input $\langle B', w\rangle$. [Recall that $D_{\text{A-DFA}}$ decides A_{DFA}]
3. *Accept* if $D_{\text{A-DFA}}$ accepts.
   Reject if not."

New element: Use conversion construction and previously constructed TM as a subroutine.

Emptiness Problem for DFAs

Let $E\text{DFA} = \left\{ \langle B \rangle \,\middle|\, B \text{ is a DFA and } L(B) = \emptyset \right\}$

Emptiness Problem for DFAs

Let $E\text{DFA} = \left\{\langle B\rangle \middle| \; B \text{ is a DFA and } L(B) = \varnothing\right\}$

Theorem: EDFA is decidable

Emptiness Problem for DFAs

Let $E\text{DFA} = \left\{\langle B\rangle \,\middle|\, B \text{ is a DFA and } L(B) = \varnothing\right\}$

Theorem: EDFA is decidable

Proof: Give TM $D_{E\text{-DFA}}$ that decides EDFA .

Emptiness Problem for DFAs

Let $E_{\text{DFA}} = \{\langle B \rangle \mid B$ is a DFA and $L(B) = \emptyset\}$

Theorem: E_{DFA} is decidable

Proof: Give TM $D_{\text{E-DFA}}$ that decides E_{DFA} .

$D_{\text{E-DFA}} =$ "On input $\langle B \rangle$ [IDEA: Check for a path from start to accept.]

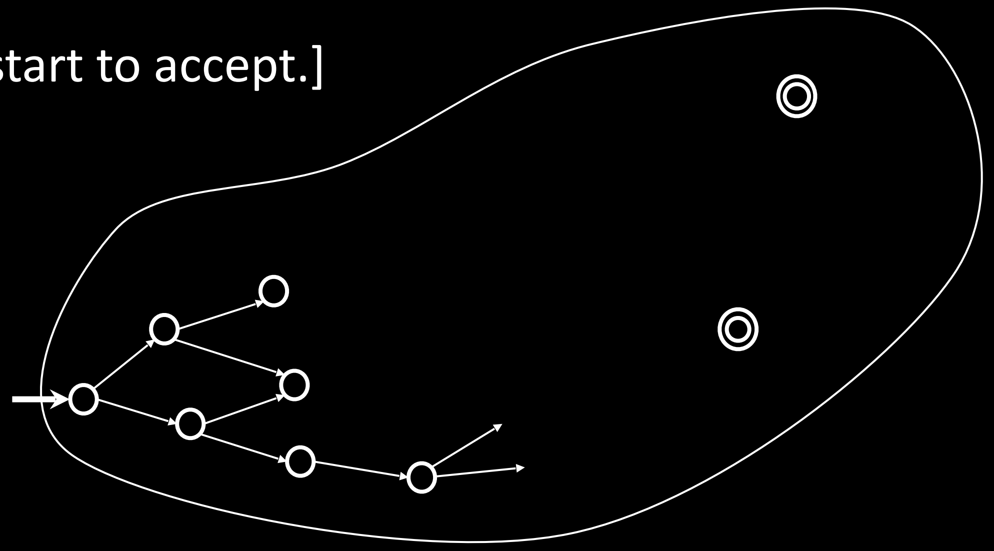
Emptiness Problem for DFAs

Let $E\text{DFA} = \{\langle B\rangle \mid B$ is a DFA and $L(B) = \varnothing\}$

Theorem: EDFA is decidable

Proof: Give TM DE−DFA that decides EDFA .

$D\text{E–DFA} =$ "On input $\langle B\rangle$ [IDEA: Check for a path from start to accept.]



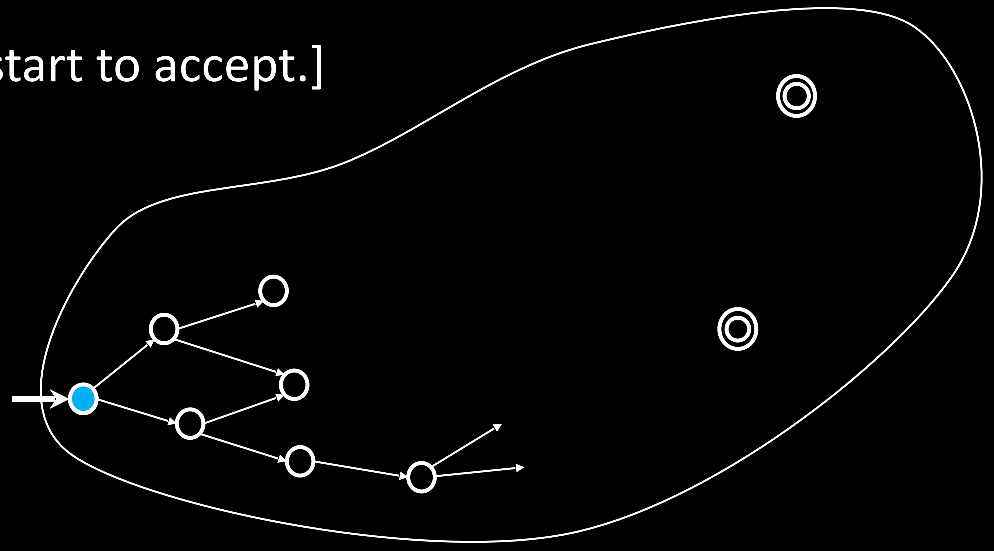
Emptiness Problem for DFAs

Let $E_{\text{DFA}} = \{\langle B \rangle \mid B$ is a DFA and $L(B) = \varnothing\}$

Theorem: E_{DFA} is decidable

Proof: Give TM $D_{\text{E-DFA}}$ that decides E_{DFA} .

$D_{\text{E–DFA}} =$ "On input $\langle B \rangle$ [IDEA: Check for a path from start to accept.]



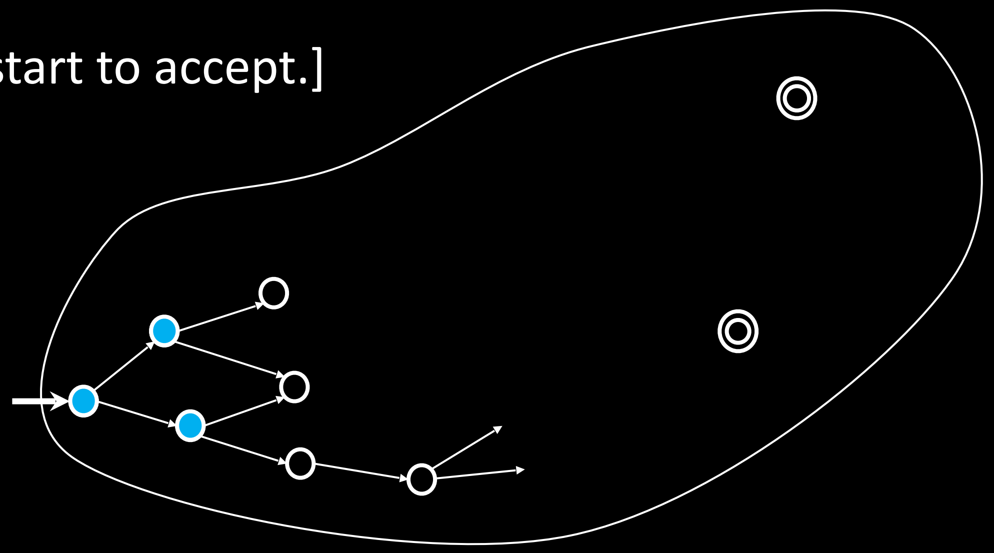
Emptiness Problem for DFAs

Let $E\text{DFA} = \{\langle B \rangle \mid B \text{ is a DFA and } L(B) = \varnothing\}$

Theorem: EDFA is decidable

Proof: Give TM DE−DFA that decides EDFA .

DE–DFA = "On input $\langle B \rangle$ [IDEA: Check for a path from start to accept.]



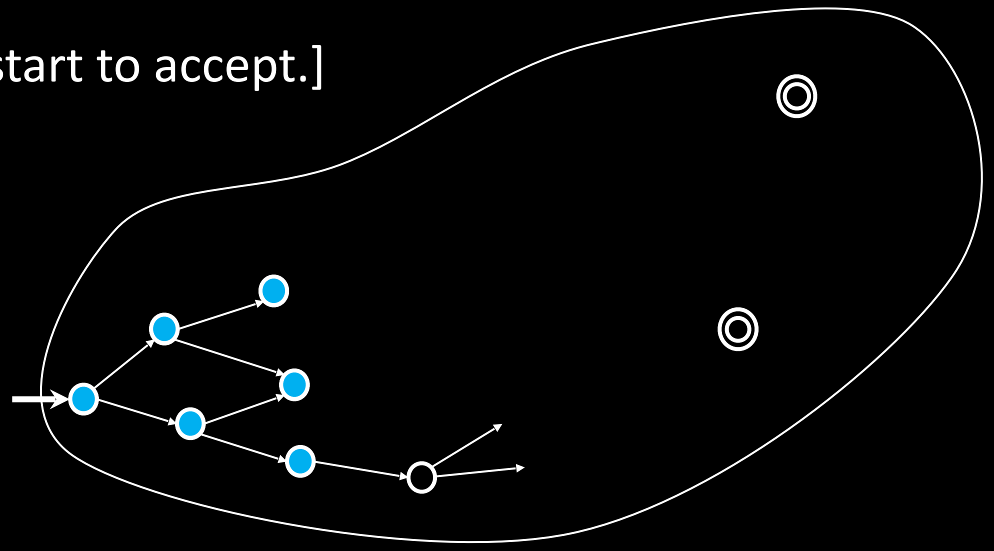
Emptiness Problem for DFAs

Let $E\text{DFA} = \{\langle B \rangle \mid B$ is a DFA and $L(B) = \varnothing\}$

Theorem: EDFA is decidable

Proof: Give TM DE−DFA that decides EDFA .

$D\text{E–DFA} =$ "On input $\langle B \rangle$ [IDEA: Check for a path from start to accept.]



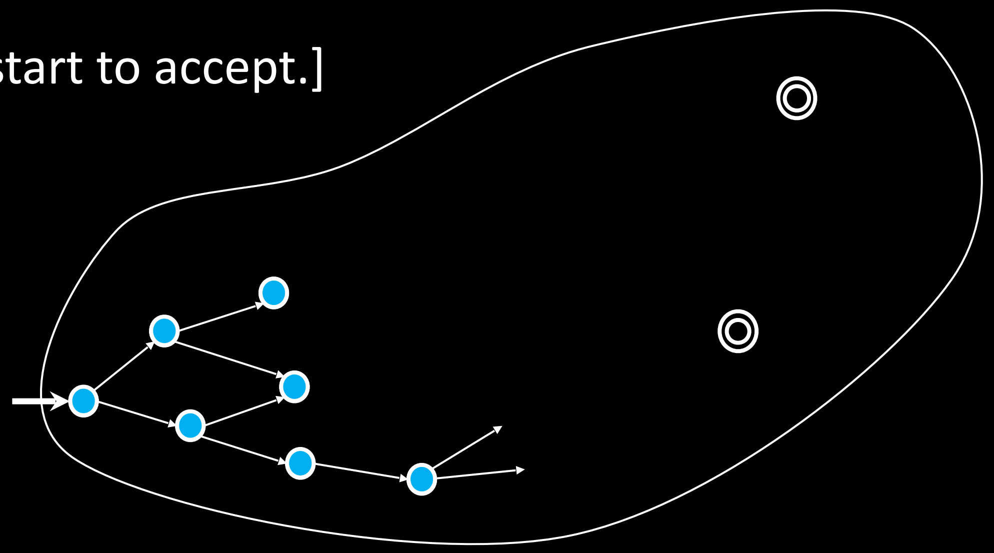
Emptiness Problem for DFAs

Let $E\text{DFA} = \{\langle B \rangle \mid B \text{ is a DFA and } L(B) = \varnothing\}$

Theorem: EDFA is decidable

Proof: Give TM DE−DFA that decides EDFA .

DE–DFA = "On input $\langle B \rangle$ [IDEA: Check for a path from start to accept.]



Emptiness Problem for DFAs

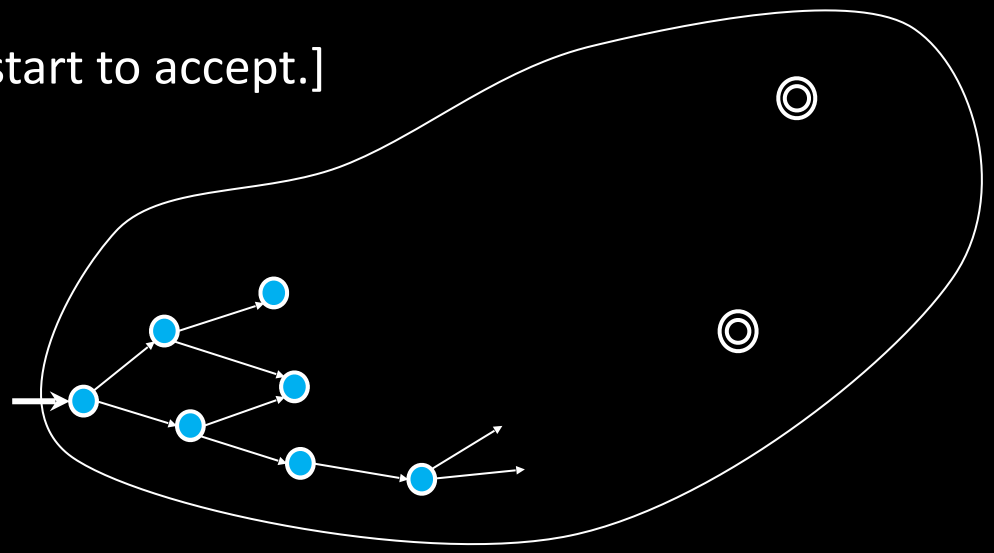
Let $E\text{DFA} = \{\langle B \rangle \mid B$ is a DFA and $L(B) = \emptyset\}$

Theorem: EDFA is decidable

Proof: Give TM DE−DFA that decides EDFA .

DE–DFA $=$ "On input $\langle B \rangle$ [IDEA: Check for a path from start to accept.]

1. Mark start state.



Emptiness Problem for DFAs

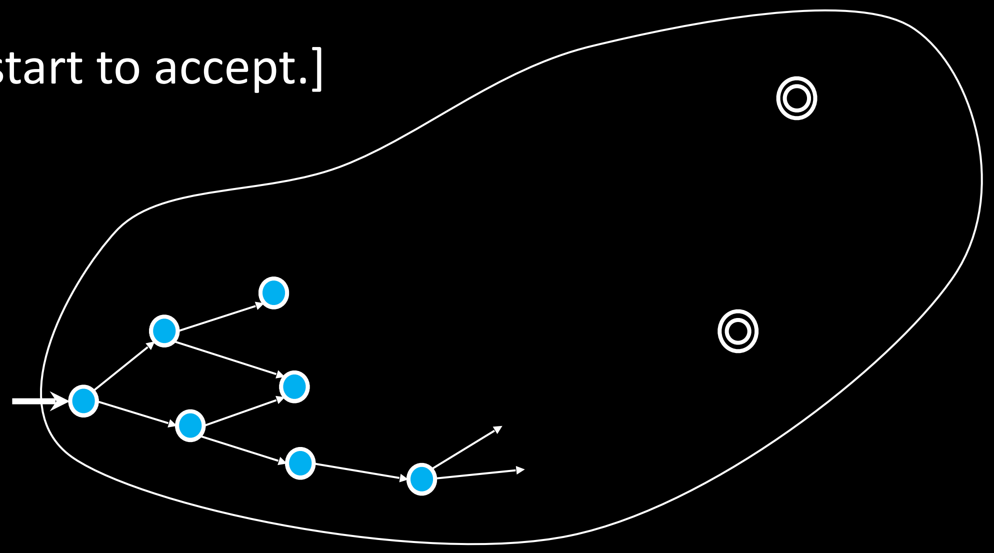
Let $E\text{DFA} = \{\langle B \rangle \mid B$ is a DFA and $L(B) = \varnothing\}$

Theorem: EDFA is decidable

Proof: Give TM DE–DFA that decides EDFA .

DE–DFA = "On input $\langle B \rangle$ [IDEA: Check for a path from start to accept.]

1. Mark start state.
2. Repeat until no new state is marked:



Emptiness Problem for DFAs

Let $E\text{DFA} = \{\langle B\rangle \mid B$ is a DFA and $L(B) = \varnothing\}$

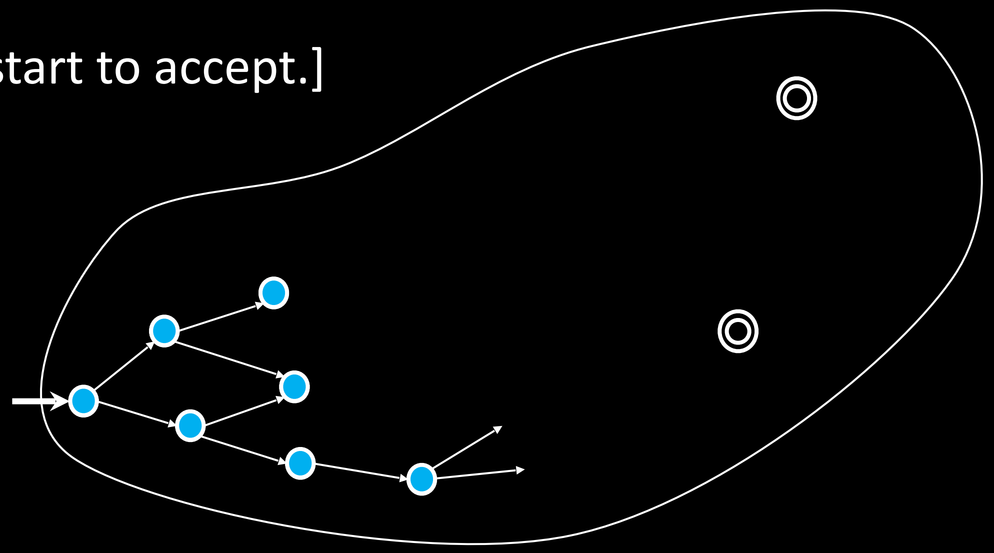
Theorem: EDFA is decidable

Proof: Give TM DE–DFA that decides EDFA .

DE–DFA = "On input $\langle B\rangle$ [IDEA: Check for a path from start to accept.]

1. Mark start state.
2. Repeat until no new state is marked:

   Mark every state that has an incoming arrow from a previously marked state.



Emptiness Problem for DFAs

Let $E_{\text{DFA}} = \left\{ \langle B \rangle \middle| \; B \text{ is a DFA and } L(B) = \varnothing \right\}$

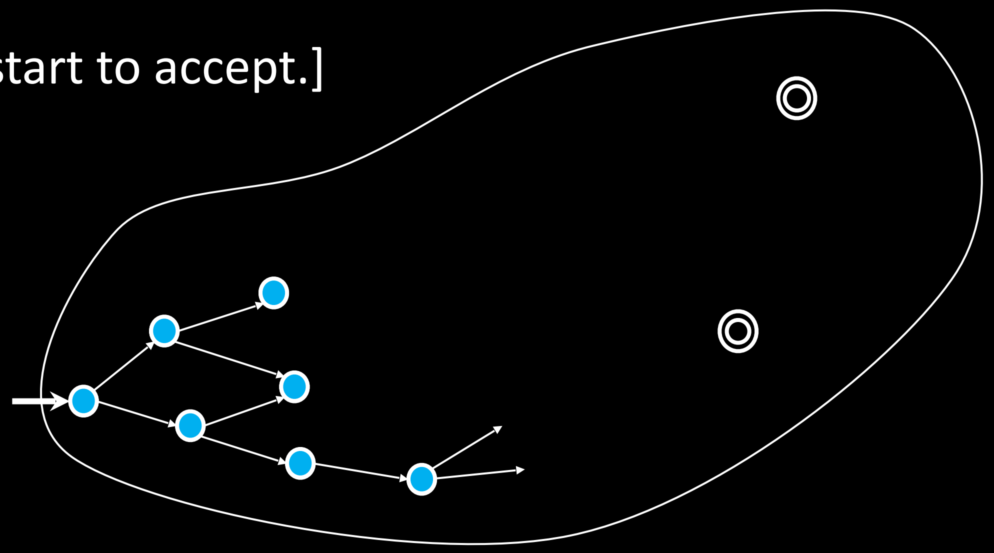
Theorem: E_{DFA} is decidable

Proof: Give TM $D_{E\text{-DFA}}$ that decides E_{DFA} .

$D_{E\text{-DFA}} =$ "On input $\langle B \rangle$ [IDEA: Check for a path from start to accept.]

1. Mark start state.
2. Repeat until no new state is marked:

   Mark every state that has an incoming arrow from a previously marked state.
3. *Accept* if no accept state is marked.
   Reject if some accept state is marked."



Emptiness Problem for DFAs

Let $E_{\text{DFA}} = \left\{ \langle B \rangle \,\middle|\, B \text{ is a DFA and } L(B) = \emptyset \right\}$

Theorem: E_{DFA} is decidable

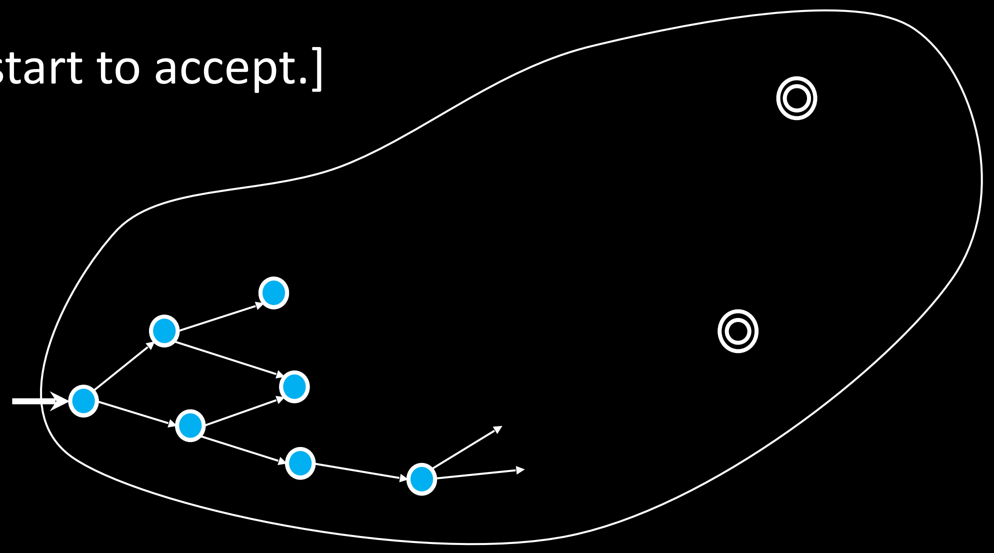
Proof: Give TM $D_{E\text{-DFA}}$ that decides E_{DFA} .

$D_{E\text{–DFA}} =$ "On input $\langle B \rangle$ [IDEA: Check for a path from start to accept.]

1. Mark start state.
2. Repeat until no new state is marked:

   Mark every state that has an incoming arrow from a previously marked state.
3. *Accept* if no accept state is marked.

   Reject if some accept state is marked."



Equivalence problem for DFAs

Let $EQ_{\text{DFA}} = \{\langle A, B\rangle \mid A$ and B are DFAs and $L(A) = L(B)\}$

Equivalence problem for DFAs

Let EQDFA $= \{\langle A, B\rangle \mid A$ and B are DFAs and $L(A) = L(B)\}$

Theorem: EQDFA is decidable

Equivalence problem for DFAs

Let EQDFA $= \{\langle A, B\rangle \mid A$ and B are DFAs and $L(A) = L(B)\}$

Theorem: EQDFA is decidable

Proof: Give TM DEQ–DFA that decides EQDFA .

Equivalence problem for DFAs

Let EQDFA $= \{\langle A, B\rangle \mid A$ and B are DFAs and $L(A) = L(B)\}$

Theorem: EQDFA is decidable

Proof: Give TM DEQ–DFA that decides EQDFA .

DEQ–DFA $=$ "On input $\langle A, B\rangle$ [IDEA: Make DFA C that accepts w where A and B disagree.]

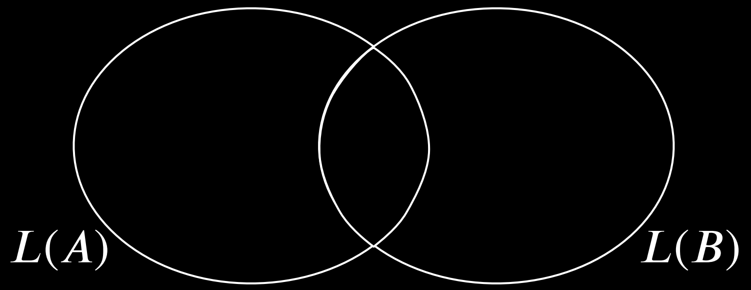
Equivalence problem for DFAs

Let EQDFA $= \{\langle A, B\rangle \mid A$ and B are DFAs and $L(A) = L(B)\}$

Theorem: EQDFA is decidable

Proof: Give TM DEQ–DFA that decides EQDFA .

DEQ–DFA $=$ "On input $\langle A, B\rangle$ [IDEA: Make DFA C that accepts w where A and B disagree.]



Symmetric difference

Equivalence problem for DFAs

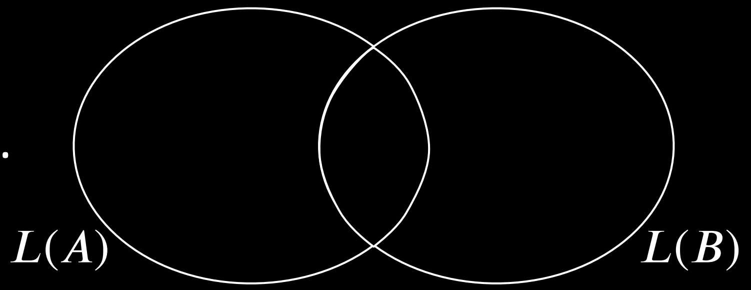
Let EQDFA $= \{\langle A, B\rangle \mid A$ and B are DFAs and $L(A) = L(B)\}$

Theorem: EQDFA is decidable

Proof: Give TM DEQ–DFA that decides EQDFA .

DEQ–DFA $=$ "On input $\langle A, B\rangle$ [IDEA: Make DFA C that accepts w where A and B disagree.]

1. Construct DFA C where $L(C) = \left(L(A) \cap \overline{L(B)}\right) \cup \left(\overline{L(A)} \cap L(B)\right)$.



Symmetric difference

Equivalence problem for DFAs

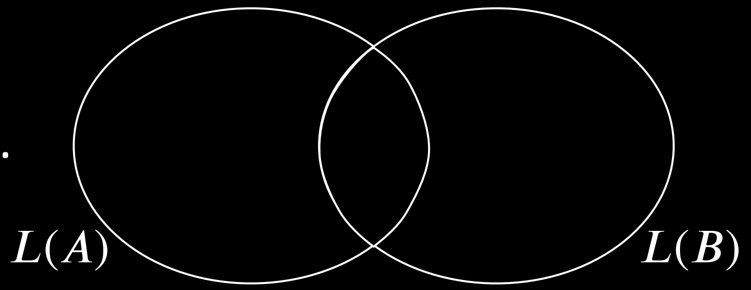
Let $EQ_{\text{DFA}} = \{\langle A, B\rangle \mid A$ and B are DFAs and $L(A) = L(B)\}$

Theorem: EQ_{DFA} is decidable

Proof: Give TM $D_{\text{EQ-DFA}}$ that decides EQ_{DFA} .

$D_{\text{EQ-DFA}} =$ "On input $\langle A, B\rangle$ [IDEA: Make DFA C that accepts w where A and B disagree.]

1. Construct DFA C where $L(C) = \left(L(A) \cap \overline{L(B)}\right) \cup \left(\overline{L(A)} \cap L(B)\right)$.
2. Run $D_{\text{E-DFA}}$ on $\langle C\rangle$.



Symmetric difference

Equivalence problem for DFAs

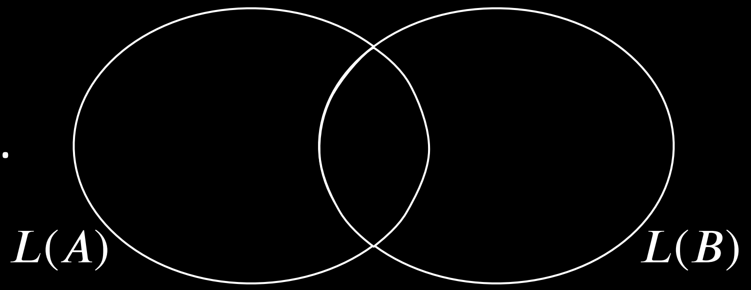
Let $EQ\text{DFA} = \{\langle A, B\rangle \mid A$ and B are DFAs and $L(A) = L(B)\}$

Theorem: EQDFA is decidable

Proof: Give TM DEQ−DFA that decides EQDFA .

$D\text{EQ–DFA} =$ "On input $\langle A, B\rangle$ [IDEA: Make DFA C that accepts w where A and B disagree.]

1. Construct DFA C where $L(C) = \left(L(A) \cap \overline{L(B)}\right) \cup \left(\overline{L(A)} \cap L(B)\right)$.
2. Run DE−DFA on $\langle C\rangle$.
3. *Accept* if DE−DFA accepts.
   Reject if DE−DFA rejects."



Symmetric difference

Equivalence problem for DFAs

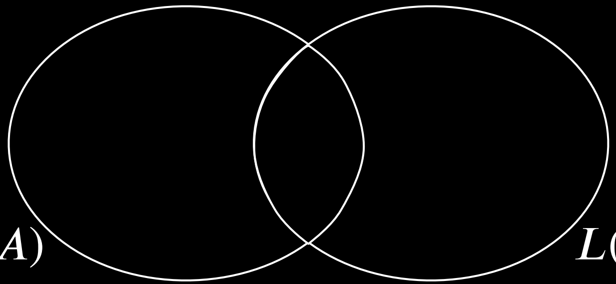
Let EQDFA $= \{\langle A, B\rangle \mid A$ and B are DFAs and $L(A) = L(B)\}$

Theorem: EQDFA is decidable

Proof: Give TM DEQ−DFA that decides EQDFA .

DEQ−DFA $=$ "On input $\langle A, B\rangle$ [IDEA: Make DFA C that accepts w where A and B disagree.]

1. Construct DFA C where $L(C) = \left(L(A) \cap \overline{L(B)}\right) \cup \left(\overline{L(A)} \cap L(B)\right)$.
2. Run DE−DFA on $\langle C\rangle$.
3. *Accept* if DE−DFA accepts.
   Reject if DE−DFA rejects."



$L(A)$ $L(B)$

Symmetric difference

Check-in 7.1

Equivalence problem for DFAs

Let $EQ_{\text{DFA}} = \{\langle A, B\rangle \mid A$ and B are DFAs and $L(A) = L(B)\}$

Theorem: EQ_{DFA} is decidable

Proof: Give TM $D_{\text{EQ-DFA}}$ that decides EQ_{DFA} .

Check-in 7.1

Let $EQ_{\text{REX}} = \{\langle R_1, R_2\rangle \mid R_1$ and R_2 are regular expressions and $L(R_1) = L(R_2)\}$

Can we now conclude that EQ_{REX} is decidable?

a) Yes, it follows immediately from things we've already shown.

b) Yes, but it would take significant additional work.

c) No, intersection is not a regular operation.

Acceptance Problem for CFGs

Let

$A\text{CFG} = \{\langle G, w\rangle \mid \quad G$ is a CFG and $w \in L(G)\}$

Acceptance Problem for CFGs

Let

$A\text{CFG} = \{\langle G, w\rangle \mid \quad G \text{ is a CFG and } w \in L(G)\}$

Theorem: ACFG is decidable

Acceptance Problem for CFGs

Let

ACFG $= \{\langle G, w\rangle \mid \quad G$ is a CFG and $w \in L(G)\}$

Theorem: ACFG is decidable

Proof: Give TM DA−CFG that decides ACFG .

Acceptance Problem for CFGs

Let

$A_{\mathsf{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG and } w \in L(G)\}$

Theorem: A_{CFG} is decidable

Proof: Give TM $D_{\mathsf{A-CFG}}$ that decides A_{CFG} .

Recall Chomsky Normal Form (CNF) only allows rules:

A → BC

B → b

Lemma 1: Can convert every CFG into CNF.
Proof and construction in book.

Lemma 2: If H is in CNF and $w \in L(H)$ then every derivation of w has $2|w| - 1$ steps.
Proof: exercise.

Acceptance Problem for CFGs

Let

ACFG $= \{\langle G, w\rangle \mid$ G is a CFG and $w \in L(G)\}$

Theorem: ACFG is decidable

Proof: Give TM DA–CFG that decides ACFG .

DA–CFG = "On input $\langle G, w\rangle$

Recall Chomsky Normal Form (CNF) only allows rules:

A → BC

B → b

Lemma 1: Can convert every CFG into CNF.
Proof and construction in book.

Lemma 2: If H is in CNF and $w \in L(H)$ then every derivation of w has $2|w| - 1$ steps.
Proof: exercise.

Acceptance Problem for CFGs

Let

$A_{\text{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG and } w \in L(G)\}$

Theorem: A_{CFG} is decidable

Proof: Give TM $D_{\text{A-CFG}}$ that decides A_{CFG}.

$D_{\text{A-CFG}} =$ "On input $\langle G, w\rangle$

1. Convert G into CNF.

Recall Chomsky Normal Form (CNF) only allows rules:

A $\rightarrow$ BC

B $\rightarrow$ b

Lemma 1: Can convert every CFG into CNF. Proof and construction in book.

Lemma 2: If H is in CNF and $w \in L(H)$ then every derivation of w has $2|w| - 1$ steps.
Proof: exercise.

Acceptance Problem for CFGs

Let

$A\text{CFG} = \{\langle G, w\rangle \mid\ G \text{ is a CFG and } w \in L(G)\}$

Theorem: ACFG is decidable

Proof: Give TM DA–CFG that decides ACFG .

DA–CFG = "On input $\langle G, w\rangle$

1. Convert G into CNF.
2. Try all derivations of length $2|w| - 1$.

Recall Chomsky Normal Form (CNF) only allows rules:

A → BC

B → b

Lemma 1: Can convert every CFG into CNF.
Proof and construction in book.

Lemma 2: If H is in CNF and $w \in L(H)$ then every derivation of w has $2|w| - 1$ steps.
Proof: exercise.

Acceptance Problem for CFGs

Let

$A_{\text{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG and } w \in L(G)\}$

Theorem: A_{CFG} is decidable

Proof: Give TM $D_{\text{A-CFG}}$ that decides A_{CFG} .

$D_{\text{A-CFG}} =$ "On input $\langle G, w\rangle$

1. Convert G into CNF.
2. Try all derivations of length $2|w| - 1$.
3. *Accept* if any generate w.
   Reject if not.

Recall Chomsky Normal Form (CNF) only allows rules:

A → BC

B → b

Lemma 1: Can convert every CFG into CNF.
Proof and construction in book.

Lemma 2: If H is in CNF and $w \in L(H)$ then every derivation of w has $2|w| - 1$ steps.
Proof: exercise.

Acceptance Problem for CFGs

Let

$A\text{CFG} = \{\langle G, w\rangle \mid \quad G \text{ is a CFG and } w \in L(G)\}$

Theorem: ACFG is decidable

Proof: Give TM DA−CFG that decides ACFG .

$D\text{A–CFG} =$ "On input $\langle G, w\rangle$

1. Convert G into CNF.
2. Try all derivations of length $2|w| - 1.$
3. *Accept* if any generate w.
   Reject if not.

Corollary: Every CFL is decidable.

Recall Chomsky Normal Form (CNF) only allows rules:

A → BC
B → b

Lemma 1: Can convert every CFG into CNF.
Proof and construction in book.

Lemma 2: If H is in CNF and $w \in L(H)$ then every derivation of w has $2|w| - 1$ steps.
Proof: exercise.

Acceptance Problem for CFGs

Let

$ACFG = \{\langle G, w\rangle \mid G$ is a CFG and $w \in L(G)\}$

Theorem: ACFG is decidable

Proof: Give TM D_{A-CFG} that decides A_{CFG} .

$D_{A-CFG} =$ "On input $\langle G, w\rangle$

1. Convert G into CNF.
2. Try all derivations of length $2|w| - 1$.
3. *Accept* if any generate w.
   Reject if not.

Corollary: Every CFL is decidable.

Proof: Let A be a CFL, generated by CFG G.

Recall Chomsky Normal Form (CNF) only allows rules:

A → BC

B → b

Lemma 1: Can convert every CFG into CNF.
Proof and construction in book.

Lemma 2: If H is in CNF and $w \in L(H)$ then every derivation of w has $2|w| - 1$ steps.
Proof: exercise.

Acceptance Problem for CFGs

Let

ACFG $= \{\langle G, w\rangle \mid \quad G$ is a CFG and $w \in L(G)\}$

Theorem: ACFG is decidable

Proof: Give TM DA–CFG that decides ACFG .

$D\mathrm{A–CFG} =$ "On input $\langle G, w\rangle$

1. Convert G into CNF.
2. Try all derivations of length $2|w|-1$.
3. *Accept* if any generate w.
   Reject if not.

Corollary: Every CFL is decidable.

Proof: Let A be a CFL, generated by CFG G.

Construct TM $M_G =$ "on input w

Recall Chomsky Normal Form (CNF) only allows rules:

A $\rightarrow$ BC

B $\rightarrow$ b

Lemma 1: Can convert every CFG into CNF.
Proof and construction in book.

Lemma 2: If H is in CNF and $w \in L(H)$ then every derivation of w has $2|w|-1$ steps.
Proof: exercise.

Acceptance Problem for CFGs

Let

$A\text{CFG} = \{\langle G, w\rangle \mid G \text{ is a CFG and } w \in L(G)\}$

Theorem: ACFG is decidable

Proof: Give TM DA–CFG that decides ACFG .

$D\text{A–CFG} =$ "On input $\langle G, w\rangle$

1. Convert G into CNF.
2. Try all derivations of length $2|w| - 1$.
3. *Accept* if any generate w.
   Reject if not.

Corollary: Every CFL is decidable.

Proof: Let A be a CFL, generated by CFG G.

Construct TM $M_G =$ "on input w

1. Run DA–CFG on $\langle G, w\rangle$.

Recall Chomsky Normal Form (CNF) only allows rules:

A $\rightarrow$ BC

B $\rightarrow$ b

Lemma 1: Can convert every CFG into CNF.
Proof and construction in book.

Lemma 2: If H is in CNF and $w \in L(H)$ then every derivation of w has $2|w| - 1$ steps.
Proof: exercise.

Acceptance Problem for CFGs

Let

$A_{\mathrm{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG and } w \in L(G)\}$

Theorem: A_{CFG} is decidable

Proof: Give TM $D_{\mathrm{A-CFG}}$ that decides A_{CFG}.

$D_{\mathrm{A-CFG}} =$ "On input $\langle G, w\rangle$

1. Convert G into CNF.
2. Try all derivations of length $2|w| - 1$.
3. *Accept* if any generate w.
   Reject if not.

Corollary: Every CFL is decidable.

Proof: Let A be a CFL, generated by CFG G.

Construct TM $M_G =$ "on input w

1. Run $D_{\mathrm{A-CFG}}$ on $\langle G, w\rangle$.
2. *Accept* if $D_{\mathrm{A-CFG}}$ accepts
   Reject if it rejects."

Recall Chomsky Normal Form (CNF) only allows rules:

A $\to$ BC

B $\to$ b

Lemma 1: Can convert every CFG into CNF.
Proof and construction in book.

Lemma 2: If H is in CNF and $w \in L(H)$ then every derivation of w has $2|w| - 1$ steps.
Proof: exercise.

Acceptance Problem for CFGs

Let

ACFG $= \{\langle G, w\rangle \mid \quad G$ is a CFG and $w \in L(G)\}$

Theorem: ACFG is decidable

Proof: Give TM DA−CFG that decides ACFG .

DA–CFG $=$ "On input $\langle G, w\rangle$

1. Convert G into CNF.
2. Try all derivations of length $2|w|-1$.
3. *Accept* if any generate w.
   Reject if not.

Corollary: Every CFL is decidable.

Proof: Let A be a CFL, generated by CFG G.

Construct TM $M_G =$ "on input w

1. Run DA−CFG on $\langle G, w\rangle$.
2. *Accept* if DA−CFG accepts
   Reject if it rejects."

Recall Chomsky Normal Form (CNF) only allows rules:

A → BC
B → b

Lemma 1: Can convert every CFG into CNF.
Proof and construction in book.

Lemma 2: If H is in CNF and $w \in L(H)$ then every derivation of w has $2|w|-1$ steps.
Proof: exercise.

Acceptance Problem for CFGs

Let

$A_{\text{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG and } w \in L(G)\}$

Theorem: A_{CFG} is decidable

Proof: Give TM $D_{\text{A-CFG}}$ that decides A_{CFG}.

$D_{\text{A-CFG}} =$ "On input $\langle G, w\rangle$

1. Convert G into CNF.
2. Try all derivations of length $2|w| - 1$.
3. *Accept* if any generate w.
   Reject if not.

Corollary: Every CFL is decidable.

Proof: Let A be a CFL, generated by CFG G.

Construct TM $M_G =$ "on input w

1. Run $D_{\text{A-CFG}}$ on $\langle G, w\rangle$.
2. *Accept* if $D_{\text{A-CFG}}$ accepts
   Reject if it rejects."

Recall Chomsky Normal Form (CNF) only allows rules:

A $\rightarrow$ BC

B $\rightarrow$ b

Lemma 1: Can convert every CFG into CNF.
Proof and construction in book.

Lemma 2: If H is in CNF and $w \in L(H)$ then every derivation of w has $2|w| - 1$ steps.
Proof: exercise.

Check-in 7.2

Acceptance Problem for CFGs

Let

ACFG $= \{\langle G, w\rangle \mid$ G is a CFG and $w \in L(G)\}$

Theorem: ACFG is decidable

Proof: Give TM DA–CFG that decides ACFG .

DA–CFG = "On input $\langle G, w\rangle$

1. Convert G into CNF.
2. Try all derivations of length $2|w| - 1$.
3. *Accept* if any generate w.

Check-in 7.2

Can we conclude that APDA is decidable?

a) Yes.

b) No, PDAs may be nondeterministic.

c) No, PDAs may not halt.

2. *Accept* if DA–CFG accepts
   Reject if it rejects."

Recall Chomsky Normal Form (CNF) only allows rules:

A → BC

B → b

Lemma 1: Can convert every CFG into CNF. Proof and construction in book.

Lemma 2: If H is in CNF and $w \in L(H)$ then every derivation of w has $2|w| - 1$ steps. Proof: exercise.

Check-in 7.2

Emptiness Problem for CFGs

Let ECFG $= \{\langle G\rangle \mid G$ is a CFG and $L(G) = \varnothing\ \}$

Emptiness Problem for CFGs

Let ECFG $= \{\langle G\rangle \mid G$ is a CFG and $L(G) = \varnothing\}$

Theorem: ECFG is decidable

Emptiness Problem for CFGs

Let ECFG $= \{\langle G\rangle \mid G$ is a CFG and $L(G) = \varnothing \}$

Theorem: ECFG is decidable

Proof:

DE−CFG $=$ “On input $\langle G\rangle$ [IDEA: work backwards from terminals]

Emptiness Problem for CFGs

Let ECFG $= \{\langle G\rangle \mid G$ is a CFG and $L(G) = \varnothing\ \}$

Theorem: ECFG is decidable

Proof:

DE–CFG $=$ "On input $\langle G\rangle$ [IDEA: work backwards from terminals]

S $\rightarrow$ R T a

R $\rightarrow$ T b

T $\rightarrow$ a

Emptiness Problem for CFGs

Let $E\text{CFG} = \{\langle G\rangle \mid G \text{ is a CFG and } L(G) = \varnothing\}$

Theorem: ECFG is decidable

Proof:

DE−CFG = "On input $\langle G\rangle$ [IDEA: work backwards from terminals]

S → R T a

R → T b

T → a

Emptiness Problem for CFGs

Let ECFG $= \{\langle G \rangle \mid G$ is a CFG and $L(G) = \emptyset\}$

Theorem: ECFG is decidable

Proof:

DE−CFG = "On input $\langle G \rangle$ [IDEA: work backwards from terminals]

S → RTa

R → Tb

T → a

Emptiness Problem for CFGs

Let ECFG $= \{\langle G\rangle \mid G$ is a CFG and $L(G) = \varnothing\ \}$

Theorem: ECFG is decidable

Proof:

DE−CFG = "On input $\langle G\rangle$ [IDEA: work backwards from terminals]

S $\rightarrow$ R T a

R $\rightarrow$ T b

T $\rightarrow$ a

Emptiness Problem for CFGs

Let ECFG $= \{\langle G \rangle \mid G$ is a CFG and $L(G) = \varnothing \}$

Theorem: ECFG is decidable

Proof:

DE−CFG = "On input $\langle G \rangle$ [IDEA: work backwards from terminals]

S $\rightarrow$ RTa

R $\rightarrow$ Tb

T $\rightarrow$ a

Emptiness Problem for CFGs

Let ECFG $= \{\langle G\rangle \mid G$ is a CFG and $L(G) = \varnothing \}$

Theorem: ECFG is decidable

Proof:

DE−CFG $=$ "On input $\langle G\rangle$ [IDEA: work backwards from terminals]

1. Mark all occurrences of terminals in G.

$S \rightarrow RTa$

$R \rightarrow Tb$

$T \rightarrow a$

Emptiness Problem for CFGs

Let $E_{\text{CFG}} = \{\langle G\rangle \mid G$ is a CFG and $L(G) = \varnothing\}$

Theorem: E_{CFG} is decidable

Proof:

$D_{E\text{-CFG}} =$ "On input $\langle G\rangle$ [IDEA: work backwards from terminals]

1. Mark all occurrences of terminals in G.
2. Repeat until no new variables are marked

$S \to RTa$

$R \to Tb$

$T \to a$

Emptiness Problem for CFGs

Let ECFG $= \{\langle G\rangle \mid G$ is a CFG and $L(G) = \varnothing\}$

Theorem: ECFG is decidable

Proof:

DE−CFG = "On input $\langle G\rangle$ [IDEA: work backwards from terminals]

1. Mark all occurrences of terminals in G.
2. Repeat until no new variables are marked

   Mark all occurrences of variable A if

   A $\rightarrow \mathrm{B}_1\mathrm{B}_2\cdots\mathrm{B}_k$ is a rule and all B_i were already marked.

S $\rightarrow$ RTa

R $\rightarrow$ Tb

T $\rightarrow$ a

Emptiness Problem for CFGs

Let ECFG $= \{\langle G\rangle \mid G$ is a CFG and $L(G) = \varnothing\}$

Theorem: ECFG is decidable

Proof:

DE−CFG = "On input $\langle G\rangle$ [IDEA: work backwards from terminals]

1. Mark all occurrences of terminals in G.
2. Repeat until no new variables are marked
   Mark all occurrences of variable A if
   A $\rightarrow \mathrm{B}_1\mathrm{B}_2\cdots\mathrm{B}_k$ is a rule and all B_i were already marked.
3. *Reject* if the start variable is marked.
   Accept if not."

S $\rightarrow$ RTa

R $\rightarrow$ Tb

T $\rightarrow$ a

Emptiness Problem for CFGs

Let $E_{\mathrm{CFG}} = \{\langle G\rangle \mid G$ is a CFG and $L(G) = \varnothing\}$

Theorem: E_{CFG} is decidable

Proof:

$D_{E\text{-CFG}} =$ "On input $\langle G\rangle$ [IDEA: work backwards from terminals]

1. Mark all occurrences of terminals in G.
2. Repeat until no new variables are marked
   Mark all occurrences of variable A if
   A $\rightarrow \mathrm{B}_1\mathrm{B}_2\cdots\mathrm{B}_k$ is a rule and all B_i were already marked.
3. *Reject* if the start variable is marked.
   Accept if not."

S $\rightarrow$ RTa
R $\rightarrow$ Tb
T $\rightarrow$ a

Equivalence Problem for CFGs

Let EQCFG $= \{\langle G, H\rangle \mid G, H$ are CFGs and $L(G) = L(H)\}$

Equivalence Problem for CFGs

Let EQCFG $= \{\langle G, H\rangle \mid G, H$ are CFGs and $L(G) = L(H)\ \}$

Theorem: EQCFG is NOT decidable

Equivalence Problem for CFGs

Let EQCFG $= \{\langle G, H\rangle \mid G, H$ are CFGs and $L(G) = L(H)\ \}$

Theorem: EQCFG is NOT decidable

Proof: Later …

Equivalence Problem for CFGs

Let EQCFG $= \{\langle G, H\rangle \mid G, H$ are CFGs and $L(G) = L(H)\}$

Theorem: EQCFG is NOT decidable

Proof: Later …

Let $AMBIG$CFG $= \{\langle G\rangle \mid G$ is an ambiguous CFG $\}$

Equivalence Problem for CFGs

Let EQCFG $= \{\langle G, H\rangle \mid G, H$ are CFGs and $L(G) = L(H)$ $\}$

Theorem: EQCFG is NOT decidable

Proof: Later …

Let $AMBIG$CFG $=$ $\{\langle G\rangle \mid$ G is an ambiguous CFG $\}$

Theorem: $AMBIG$CFG is NOT decidable

Equivalence Problem for CFGs

Let EQCFG $= \{\langle G, H\rangle \mid G, H$ are CFGs and $L(G) = L(H)\ \}$

Theorem: EQCFG is NOT decidable

Proof: Later …

Let $AMBIG$CFG $= \{\langle G\rangle \mid G$ is an ambiguous CFG $\}$

Theorem: $AMBIG$CFG is NOT decidable

Proof: Homework.

Equivalence Problem for CFGs

Let EQCFG $= \{\langle G, H\rangle \mid G, H$ are CFGs and $L(G) = L(H)\ \}$

Theorem: EQCFG is NOT decidable
Proof: Later …

Let $AMBIG$CFG $= \{\langle G\rangle \mid G$ is an ambiguous CFG $\}$

Theorem: $AMBIG$CFG is NOT decidable
Proof: Homework.

Check-in 7.3

Equivalence Problem for CFGs

Let EQCFG $= \{\langle G, H\rangle \mid G, H$ are CFGs and $L(G) = L(H) \}$

Theorem: EQCFG is NOT decidable

Proof: Later …

Let $AMBIG$CFG $= \{\langle G\rangle \mid G$ is an ambiguous CFG $\}$

Check-in 7.3

Why can’t we use the same technique we used to show EQDFA is decidable to show that EQCFG is decidable?

a) Because CFGs are generators and DFAs are recognizers.

b) Because CFLs are closed under union.

c) Because CFLs are not closed under complementation and intersection.

Check-in 7.3

Acceptance Problem for TMs

Let ATM $= \{\langle M, w\rangle \mid M$ is a TM and M accepts $w\}$

Acceptance Problem for TMs

Let ATM $= \{\langle M, w\rangle \mid M$ is a TM and M accepts $w\}$

Theorem: ATM is not decidable

Acceptance Problem for TMs

Let ATM $= \{\langle M, w\rangle \mid M$ is a TM and M accepts $w\}$

Theorem: ATM is not decidable

Proof: Later ....

Acceptance Problem for TMs

Let ATM $= \{\langle M, w\rangle \mid M$ is a TM and M accepts $w\}$

Theorem: ATM is not decidable
Proof: Later ....

Theorem: ATM is T-recognizable

Acceptance Problem for TMs

Let ATM $= \{\langle M, w\rangle \mid M$ is a TM and M accepts $w\}$

Theorem: ATM is not decidable

Proof: Later ....

Theorem: ATM is T-recognizable

Proof: The following TM U recognizes ATM

Acceptance Problem for TMs

Let ATM $= \{\langle M, w \rangle \mid M$ is a TM and M accepts $w\}$

Theorem: ATM is not decidable

Proof: Later ….

Theorem: ATM is T-recognizable

Proof: The following TM U recognizes ATM

$U =$ “On input $\langle M, w \rangle$

Acceptance Problem for TMs

Let ATM $= \{\langle M, w\rangle \mid M$ is a TM and M accepts $w\}$

Theorem: ATM is not decidable

Proof: Later ….

Theorem: ATM is T-recognizable

Proof: The following TM U recognizes ATM

$U =$ "On input $\langle M, w\rangle$

1. Simulate M on input w.

Acceptance Problem for TMs

Let ATM $= \{\langle M, w\rangle \mid$ M is a TM and M accepts $w\}$

Theorem: ATM is not decidable

Proof: Later ….

Theorem: ATM is T-recognizable

Proof: The following TM U recognizes ATM

$U =$ "On input $\langle M, w\rangle$

1. Simulate M on input w.
2. *Accept* if M halts and accepts.

Acceptance Problem for TMs

Let ATM $= \{\langle M, w\rangle \mid$ M is a TM and M accepts $w\}$

Theorem: ATM is not decidable
Proof: Later ….

Theorem: ATM is T-recognizable
Proof: The following TM U recognizes ATM

$U =$ "On input $\langle M, w\rangle$

1. Simulate M on input w.
2. *Accept* if M halts and accepts.
3. *Reject* if M halts and rejects.

Acceptance Problem for TMs

Let ATM $= \{\langle M, w\rangle \mid M$ is a TM and M accepts $w\}$

Theorem: ATM is not decidable

Proof: Later ....

Theorem: ATM is T-recognizable

Proof: The following TM U recognizes ATM

$U =$ "On input $\langle M, w\rangle$

1. Simulate M on input w.
2. *Accept* if M halts and accepts.
3. *Reject* if M halts and rejects.
4. *Reject* if M never halts."

Acceptance Problem for TMs

Let ATM $= \{\langle M, w\rangle \mid M$ is a TM and M accepts $w\}$

Theorem: ATM is not decidable

Proof: Later ….

Theorem: ATM is T-recognizable

Proof: The following TM U recognizes ATM

$U =$ "On input $\langle M, w\rangle$

1. Simulate M on input w.
2. *Accept* if M halts and accepts.
3. *Reject* if M halts and rejects.
4. ~~*Reject* if M never halts.~~" Not a legal TM action.

Acceptance Problem for TMs

Let $A_{TM} = \{\langle M, w\rangle \mid M$ is a TM and M accepts $w\}$

Theorem: A_{TM} is not decidable

Proof: Later ….

Theorem: A_{TM} is T-recognizable

Proof: The following TM U recognizes A_{TM}

$U =$ "On input $\langle M, w\rangle$

1. Simulate M on input w.
2. *Accept* if M halts and accepts.
3. *Reject* if M halts and rejects.
4. ~~*Reject* if M never halts.~~" Not a legal TM action.

Turing's original "Universal Computing Machine"

Acceptance Problem for TMs

Let ATM $= \{\langle M, w\rangle \mid M$ is a TM and M accepts $w\}$

Theorem: ATM is not decidable

Proof: Later ….

Theorem: ATM is T-recognizable

Proof: The following TM U recognizes ATM

$U =$ “On input $\langle M, w\rangle$

1. Simulate M on input w.
2. *Accept* if M halts and accepts.
3. *Reject* if M halts and rejects.
4. ~~*Reject* if M never halts.~~” Not a legal TM action.

Turing’s original “Universal Computing Machine”

Von Neumann said U inspired the concept of a stored program computer.

Acceptance Problem for TMs

Let ATM $= \{\langle M, w\rangle \mid M$ is a TM and M accepts $w\}$

Theorem: ATM is not decidable

Proof: Later ….

Theorem: ATM is T-recognizable

Proof: The following TM U recognizes ATM

$U =$ "On input $\langle M, w\rangle$

1. Simulate M on input w.
2. *Accept* if M halts and accepts.
3. *Reject* if M halts and rejects.
4. ~~*Reject* if M never halts.~~" Not a legal TM action.

Turing's original "Universal Computing Machine"

Von Neumann said U inspired the concept of a stored program computer.