CS 181 - Decidability

Alexander A. Sherstov

UCLA

Buridan's donkey

Part 1

Basic notions

Definition

A Turing machine M **decides** language L if:

- M halts in q_{accept} on every input $x \in L$; and
- M halts in q_{reject} on every input $x \notin L$.

Definitions

Definition

A Turing machine M **decides** language L if:

- M halts in q_{accept} on every input $x \in L$; and
- M halts in q_{reject} on every input $x \notin L$.

Such M is called a **decider** for L, and L is called **decidable.**

Deciding L = recognizing L + halting on every input.

Examples

We showed earlier that the following languages are decidable:

- $\{w\#w : w \in \{0,1\}^*\}$
- $\{0^{2^n} : n \geq 0\}$
- $\{a^n b^{kn} : n, k \geq 1\}$
- $\{w_1\#w_2\#\cdots\#w_k : w_1, \ldots, w_k \in \{0,1\}^* \text{ pairwise distinct}\}$

Examples

We showed earlier that the following languages are decidable:

- $\{w\#w : w \in \{0,1\}^*\}$
- $\{0^{2^n} : n \geq 0\}$
- $\{a^n b^{kn} : n, k \geq 1\}$
- $\{w_1\#w_2\#\cdots\#w_k : w_1, \ldots, w_k \in \{0,1\}^* \text{ pairwise distinct}\}$

Today, we will look at languages that encode various **computational problems** other than text processing.

Part 2

Problems about automata

Problems about automata

As a finite discrete-mathematical object, every DFA can be encoded as a binary string!

Problems about automata

As a finite discrete-mathematical object, every DFA can be encoded as a binary string!

- Fix a DFA: $(Q, \Sigma, \delta, q_0, F)$

Problems about automata

As a finite discrete-mathematical object, every DFA can be encoded as a binary string!

- Fix a DFA: $(Q, \Sigma, \delta, q_0, F)$
- Represent the states by consecutive strings:

$$\overbrace{00\ldots00}^{\lceil \log |Q| \rceil}, \overbrace{00\ldots01}^{\lceil \log |Q| \rceil}, \ldots$$

Problems about automata

As a finite discrete-mathematical object, every DFA can be encoded as a binary string!

- Fix a DFA: $(Q, \Sigma, \delta, q_0, F)$
- Represent the states by consecutive strings:

$$\overbrace{00\ldots00}^{\lceil \log |Q| \rceil}, \overbrace{00\ldots01}^{\lceil \log |Q| \rceil}, \ldots$$

- Represent the symbols by consecutive strings:

$$\overbrace{00\ldots00}^{\lceil \log |\Sigma| \rceil}, \overbrace{00\ldots01}^{\lceil \log |\Sigma| \rceil}, \ldots$$

Complete encoding

0000010111000011100

00000 0000 00111
00000 0001 00101
⋮ ⋮ ⋮
01110 1100 00001
01010 00011 00101 ...

Encoding explained

$$\overbrace{00000}^{\lceil \log|Q| \rceil}\overbrace{10111}^{\text{highest state in } Q}\overbrace{0000}^{\lceil \log|\Sigma| \rceil}\overbrace{11100}^{\text{highest symbol in } \Sigma}$$

00000 0000 00111
00000 0001 00101
⋮ ⋮ ⋮
01110 1100 00001

δ in lex order

01010 00011 00101 ...

F

Encoding other objects

Can similarly encode regular expressions, NFAs, PDAs, CFGs, TMs.

Encoding other objects

Can similarly encode regular expressions, NFAs, PDAs, CFGs, TMs.

Definition

- $\langle A \rangle$ = the standard binary encoding of the object A
- $L(M)$ = the language recognized by the device M

DFA acceptance problem

Problem

Decide $L_1 = \{\langle D, w\rangle : \text{the DFA } D \text{ accepts the string } w\}$.

Solution to DFA acceptance problem

- Reject if encoding is invalid.

Solution to DFA acceptance problem

- Reject if encoding is invalid.
- Initialize tapes:

$\langle D \rangle$	**encoding of *D***
$\langle q_0 \rangle$	**current state of *D***
$\langle w \rangle$	**input to *D***

Solution to DFA acceptance problem

- Reject if encoding is invalid.
- Initialize tapes:

$\langle D \rangle$	**encoding of *D***
$\langle q_0 \rangle$	**current state of *D***
$\langle w \rangle$	**input to *D***

- Until end of w is reached:
  - scan description of D for next action
  - update D's state and advance to next symbol of w

Solution to DFA acceptance problem

- Reject if encoding is invalid.
- Initialize tapes:

$\langle D \rangle$	**encoding of *D***
$\langle q_0 \rangle$	**current state of *D***
$\langle w \rangle$	**input to *D***

- Until end of w is reached:
  - scan description of D for next action
  - update D's state and advance to next symbol of w
- Accept iff D's current state is in F.

Other problems about automata

Problem

Decide $L_2 = \{\langle N, w\rangle : \text{the NFA } N \text{ accepts the string } w\}$.

Problem

Decide $L_2 = \{\langle N, w\rangle : \text{the NFA } N \text{ accepts the string } w\}$.

Solution

- Convert N to an equivalent DFA D.
- Check whether D accepts w.

Other problems about automata

Problem

Decide $L_3 = \{\langle R, w\rangle$: regular expression R generates string $w\}$.

Other problems about automata

Problem

Decide $L_3 = \{\langle R, w \rangle : \text{regular expression } R \text{ generates string } w\}$.

Solution

- Convert R to an equivalent NFA N.
- Check whether N accepts w.

Other problems about automata

Problem

Decide $L_4 = \{\langle N \rangle : \ N \text{ is an NFA with } L(N) = \varnothing\}$.

Other problems about automata

Problem

Decide $L_4 = \{\langle N \rangle : \; N \text{ is an NFA with } L(N) = \varnothing\}$.

Solution

View N as a graph. Check whether an accept state is reachable from start state.

Other problems about automata

Problem

Decide $L_5 = \{\langle D', D'' \rangle : \ D', D'' \text{ are DFAs with } L(D') = L(D'')\}$.

Other problems about automata

Problem

Decide $L_5 = \{\langle D', D''\rangle : \ D', D'' \text{ are DFAs with } L(D') = L(D'')\}$.

Solution

- Construct DFA D for $L(D') \oplus L(D'')$ using Cartesian product.
- Check whether $L(D) = \varnothing$.

Other problems about automata

Problem

Decide $L_5 = \{\langle D', D'' \rangle : \ D', D'' \text{ are DFAs with } L(D') = L(D'')\}$.

Solution

- Construct DFA D for $L(D') \oplus L(D'')$ using Cartesian product.
- Check whether $L(D) = \varnothing$.

Similar: equivalence testing for NFAs and regular expressions.

Other problems about automata

Problem

On input a DFA D, decide if D accepts a string of even length.

Other problems about automata

Problem

On input a DFA D, decide if D accepts a string of even length.

Solution

- Construct DFA D' for $L(D) \cap (\Sigma\Sigma)^*$ using Cartesian product.
- Check whether $L(D') \neq \varnothing$.

Other problems about automata

Problem

On input a DFA D, decide if D accepts w^R whenever D accepts w.

Other problems about automata

Problem

On input a DFA D, decide if D accepts w^R whenever D accepts w.

Solution

- Construct DFA D' for $L(D)^R$.
- Check whether $L(D') = L(D)$.

Other problems about automata

Problem

On input a DFA D, decide whether $L(D)$ is infinite.

Other problems about automata

Problem

On input a DFA D, decide whether $L(D)$ is infinite.

Solution

- Let $k =$ number of states in D.
- By pumping lemma, $L(D)$ is infinite iff D accepts a string of length $\geq k$.

Other problems about automata

Problem

On input a DFA D, decide whether $L(D)$ is infinite.

Solution

- Let k = number of states in D.
- By pumping lemma, $L(D)$ is infinite iff D accepts a string of length $\geq k$.
- Build a DFA D' for $L(D) \cap \Sigma^k \Sigma^*$ using Cartesian product.
- Check whether $L(D') \neq \varnothing$.

Part 3

Problems about CFGs and PDAs

Problems about context-free grammars

Theorem

Let $G = (V, \Sigma, R, S)$ *be a context-free grammar. If* $L(G) \neq \varnothing$, *then some* $w \in L(G)$ *has a parse tree of depth* $\leq |V|$.

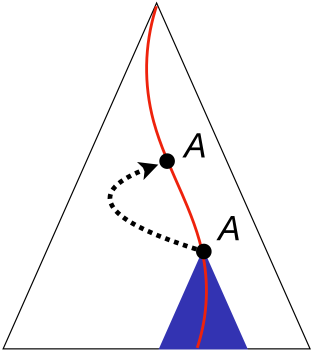
Problems about context-free grammars

Theorem

Let $G = (V, \Sigma, R, S)$ be a context-free grammar. If $L(G) \neq \varnothing$, then some $w \in L(G)$ has a parse tree of depth $\leq |V|$.

Proof.

Take smallest parse tree for G. If depth $> |V|$, then some path repeats a variable, so smaller tree exists! □



Problems about context-free grammars

Problem

On input a CFG G, decide if $L(G) \neq \varnothing$.

Problems about context-free grammars

Problem

On input a CFG G, decide if $L(G) \neq \varnothing$.

Solution

- Let $v =$ number of variables in G.
- Exhaustively search for a valid parse tree of depth $\leq v$.
- Accept iff such tree is found.

Problems about context-free grammars

Problem

On input a CFG G and string w, decide if G generates w.

Problems about context-free grammars

Problem

On input a CFG G and string w, decide if G generates w.

Solution

- Convert G to an equivalent PDA P.
- Construct PDA P' for $L(P) \cap \{w\}$ using Cartesian product.
- Convert P' to an equivalent CFG G'.
- Check whether $L(G') \neq \varnothing$.

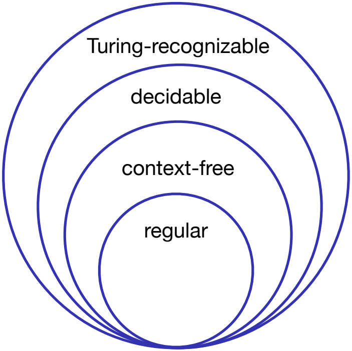
Corollary

Every context-free language is decidable.

Problems about context-free grammars

Corollary

Every context-free language is decidable.



Problems about pushdown automata

Problem

On input a PDA P, decide if P has an unreachable state.

Problem

On input a PDA P, decide if P has an unreachable state.

Solution

- Let $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$.
- For each $q \in Q$, define $P_q = (Q, \Sigma, \Gamma, \delta, q_0, \{q\})$.

Problems about pushdown automata

Problem

On input a PDA P, decide if P has an unreachable state.

Solution

- Let $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$.
- For each $q \in Q$, define $P_q = (Q, \Sigma, \Gamma, \delta, q_0, \{q\})$.
- Convert each P_q to a CFG and check if $L(P_q) \neq \varnothing$.
- Accept iff $L(P_q) \neq \varnothing$ for all $q \in Q$.

Problems about pushdown automata

Problem

On input a DFA D, decide if D accepts a palindrome.

Problems about pushdown automata

Problem

On input a DFA D, decide if D accepts a palindrome.

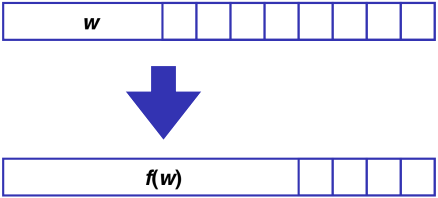
Solution

- Let P = PDA that recognizes palindromes.
- Construct PDA P' for $L(P) \cap L(D)$ using Cartesian product.
- Convert P' to an equivalent CFG G'.
- Check whether $L(G') \neq \varnothing$.

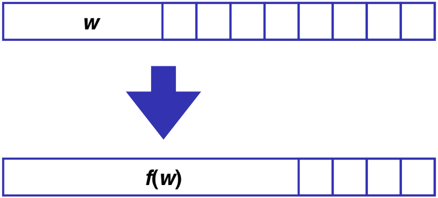
Part 4

Computation of functions

Computing a function



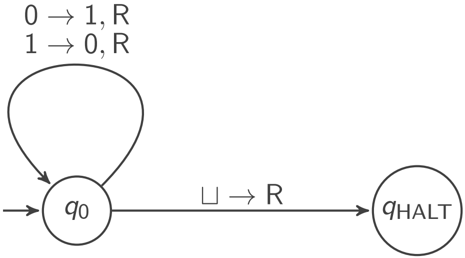
Computing a function



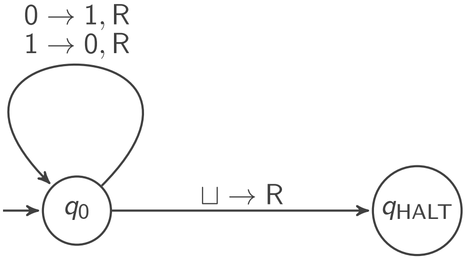
Definition

Let $f : \Sigma^* \to \Sigma^*$. Turing machine M computes f iff when started on input w, TM M eventually enters q_{HALT} with $f(w)$ written on the tape.

Computing a function

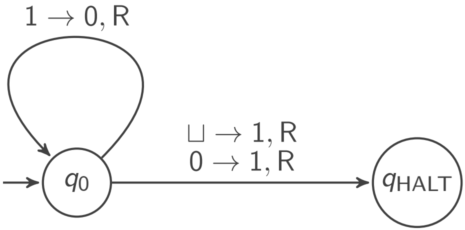


Computing a function

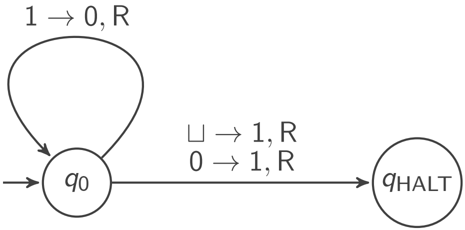


Computes componentwise complement.

Computing a function



Computing a function



Increments binary string, written starting with least significant bit.