COMPSCI 501: Formal Language Theory

Lecture 39: Review

Marius Minea
marius@cs.umass.edu

University of Massachusetts Amherst

1 May 2019

Automata and Regular Languages

Interesting questions (accept, empty, equivalence) decidable

Useful argument: convert to DFA, minize $\implies$ unique form.

Representative problem: ALL_{NFA}

$\overline{ALL_{\text{NFA}}}$: in PSPACE
nondeterministically guess rejected string
simulate, NFA in $\leq |Q|$ states at each step (PSPACE)
Does this imply $\in$ NP? No. (unknown whether NP, or coNP)

Useful idea: cross-product (shuffle, suffix languages),
also use for PDAs.

Context Free Grammars and Pushdown Automata

Chomsky Normal Form: simplify, bounds on derivation complexity
$A \to BC$, $A \to a$, $(S \to \varepsilon)$

any derivation has $2|w| - 1$ steps
e.g., decide A_{CFG}, try all derivations from S
$A_{\text{CFG}} \in$ P, dynamic programming (all terminals generating any substring)

CFG / PDA Equivalence proof:
$\Rightarrow$: nondet. choose rule, push nonterminals, match/pop terminals
$\Leftarrow$: nonterminal for each PDA state pair

Closure properties:
yes: union, concatenation, Kleene star
no: intersection, complement (only DCFL)

Pumping Lemmas

Regular: any language string with $|s| \geq p$ can be divided into three pieces, $s = xyz$, with the conditions
1. $xy^iz \in A$ for any $i \geq 0$
2. $|y| > 0$
3. $|xy| \leq p$

Context-Free: any language string with $|s| \geq p$ can be divided into five pieces, $s = uvxyz$, with the conditions\ 1. $uv^ixy^iz \in A$ for any $i \geq 0$
2. $|vy| > 0$
3. $|vxy| \leq p$

Pump *up or down* (both useful)

Unidirectional:
if pumped string not in language, is not regular/context-free
there are other languages for which pumping holds
e.g., $ca^nb^n \bigcup c^k(a+b)^*$, $k \neq 1$ nonregular, pumpable

Turing Machines, Recognizability, Decidability

A Turing recognizable (by M_1) and $\bar{A}$ Turing recognizable (by M_2)
$\implies$ A decidable

run M_1 and M_2 in lockstep, see which halts first

Enumerating

Turing-recognizable $\Leftrightarrow$ (recursively) enumerable

run for k steps on $s_1, s_2, \ldots s_k$ (dovetailing)

Decidable $\Leftrightarrow$ enumerable in lexicographical order

Important: carefully consider language of given problem:
Is is $\{\langle M, w\rangle \mid ...\}$ or just $\{\langle M\rangle \mid ...\}$?
If the latter, can't say "run M on w and ..." (what is the input w?)

Problems for Recognizers

Acceptance:
$A_M = \{\langle M, w\rangle | M$ is a machine that accepts $w\}$

Emptiness
$E_M = \{\langle M\rangle | M$ is a machine with $L(M) = \emptyset\}$

Universality
$ALL_M = \{\langle M\rangle | M$ is a machine with $L(M) = \Sigma^*\}$

Equivalence
$EQ_M = \{\langle A, B\rangle | A, B$ are machines with $L(A) = L(B)\}$

All decidable for DFA/NFA/REX (convert to minimized DFA)

A_{CFG}, E_{CFG} decidable, ALL_{CFG}, EQ_{CFG} not decidable

A_{TM}, $HALT_{\text{TM}}$, E_{TM}, etc. not decidable

Proving Undecidability of a Language L

Diagonalization (directly)

e.g., for proving A_{TM} undecidable

Reducing from A_{TM}. Example: E_{TM}.

Assume decider D for E_{TM}, build decider for A_{TM}.

Construct a TM M_1 that will either have an empty language or not, depending on whether M accepts w.
won't ever run M_1, but feed as input to D

On input x:
if $x \neq w$, *reject*
otherwise, run A on w $(= x)$, *accept* if A does

Thus, $L(M_1) = \{w\}$ if A accepts w, $\emptyset$ otherwise
Could use D to decide A_{TM}.

More General: Rice's Theorem

Let P be a nontrivial property of a Turing machine:
A *language property*, $L(M_1) = L(M_2) \rightarrow (P(M_1) \leftrightarrow P(M_2))$
At least one TM has this property.

Then P is undecidable.

Let *MP* a Turing machine with that property.
(Assume language nonempty, else pick complement).

Construct a decider for A_{TM}.

On input $\langle M, w\rangle$, construct TM C as follows:
on input x:
run M on w (*reject* or run forever like M)
if *accept*, run *MP* on x, return result

C has same language as *MP* (if M accepts w) or empty language.
$\Longrightarrow$ could use decider for P to decide A_{TM}.

Proving a Language is not Turing-recognizable

Similar idea, but reduce from $\overline{A_{\mathsf{TM}}}$.

EQ_{TM} not Turing-recognizable nor co-Turing-recognizable.

On input $\langle M, w\rangle$, construct two machines:
$M_\emptyset$: rejects any input / M_{all}: accepts any input
M_w: accept all/none, according to run of M on w

$M_\emptyset$ not EQ M_w iff M accepts w: $A_{\mathsf{TM}} \leq_{\mathsf{m}} \overline{EQ_{\mathsf{TM}}}$, $\overline{A_{\mathsf{TM}}} \leq_{\mathsf{m}} EQ_{\mathsf{TM}}$

M_{all} EQ M_w iff M accepts w: $A_{\mathsf{TM}} \leq_{\mathsf{m}} EQ_{\mathsf{TM}}$, $\overline{A_{\mathsf{TM}}} \leq_{\mathsf{m}} \overline{EQ_{\mathsf{TM}}}$

Mapping reduction: $f(\langle M, w\rangle) = \langle M_\emptyset, M_w\rangle$ or $\langle M_{all}, M_w\rangle$

Reduction via Computation Histories

Linear Bounded Automata: A_{LBA} decidable (finite number of configurations), but E_{LBA} is not.

Set of accepting TM computation histories of a TM checkable by an LBA.

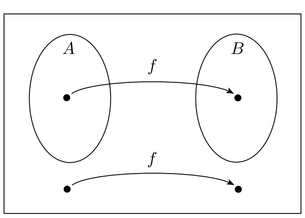
Another use: all strings that are *not* accepting computation histories on a string w.

- can generate with a PDA / CFG $\Longrightarrow$ ALL_{CFG}= undecidable (deciding $\neq \Sigma^* \Leftrightarrow$ deciding A_{TM})
- can generate via extended regular expressions
  use to prove: equivalence of extended regular expressions with exponentiation is EXPSPACE-hard.

Mapping Reducibility

Def. A function $f : \Sigma^* \rightarrow \Sigma^*$ is a **computable function** if some Turing machine M, on input w, *halts* with just $f(w)$ on tape.

Def. A language A is **mapping reducible** to language B (written $A \leq_{\mathsf{m}} B$) if there is a *computable function* $f : \Sigma^* \rightarrow \Sigma^*$ where for every w, $w \in A \Leftrightarrow f(w) \in B$



YES for A means YES for B
NO for A means NO for B

Using Mapping Reducibility

Decidability

If $A \leq_{\mathsf{m}} B$ and B is decidable, then A is decidable.

If $A \leq_{\mathsf{m}} B$ and A is undecidable then B is undecidable.

Turing-recognizability

If $A \leq_{\mathsf{m}} B$ and B is Turing-recognizable, then A is Turing-recognizable.

If $A \leq_{\mathsf{m}} B$ and A is not Turing-recognizable then B is not Turing-recognizable.

Recursion Theorem

A TM can obtain and execute its own description.

Use: e.g., in proofs by contradiction (do something else than description says)

e.g. assume A_{TM} has a decider H
Construct a TM B:
On input w:

1. Obtain own description $\langle B \rangle$
2. Run H on input $\langle B, w \rangle$
3. Do the opposite of H (accept/reject)

Descriptive Complexity

The **minimal description** of a binary string x is the shortest string $\langle M, w \rangle$ where M halts on input w with x on tape.

The **descriptive complexity** (Kolmogorov complexity) is the length of the mininmal description: $\mathsf{K}(x) = |d(x)|$

Def.: A string x is c-**compressible** if $\mathsf{K}(x) \leq |x| - c$.

incompressible = not 1-compressible.
Most strings are close to incompressible.

Incompressible strings are undecidable.

Can only enumerate a finite subset.

Polynomial Verifiers and NP

Def. A **verifier** for a language A is an algorithm V, where

$$A = \{w \mid V \text{ accepts } \langle w, c \rangle \text{ for some string } c\}$$

A **polynomial-time verifier** runs polynomial in the length of v.

A language is **polynomially verifiable** if it has a polynomial time verifier.

NP is the class of languages that have polynomial-time verifiers.

equivalent: A language is in NP iff it is decided by some nondeterministic polynomial time Turing machine

Assymmetry: **Proving** (witness) $\neq$ **Disproving** (no witness?)

Def.: A is in coNP if $\bar{A}$ in NP. P $\subseteq$ NP $\cap$ coNP

NP-Completeness

Def. A *language* B is NP-complete iff
1. B is in NP
2. B is NP-hard: for any A in NP, we have $A \leq_{\mathsf{P}} B$

Important: reduce **from**:

- to prove B is NP-hard, show $C \leq_{\mathsf{P}} B$ for NP-complete C
  reduce *known* NP-complete problem C to target B
  reduce target problem B *from* NP-complete problem C
- If B is NP-complete and $B \in$ P, then P = NP

All NP-complete problems are polynomially reducible to one another (the hardest problems in NP)

Time Complexity

Time complexity class TIME$(t(n))$ = all languages that are decidable by an $O(t(n))$ (deterministic, single-tape) Turing machine.

A $t(n)$ **multitape** TM has an equivalent $O(t^2(n))$ single-tape TM.

multi-tape *polynomial* $\Rightarrow$ single-tape *polynomial*

Every $t(n)$ **nondeterministic** TM has an equivalent $2^{O(t(n))}$ deterministic single-tape TM.

nondeterministic *polynomial* $\Rightarrow$ single-tape *exponential*

Space Complexity

Savitch's Theorem

For any function $f : \mathbb{N} \to \mathbb{R}^+$, where $f(n) \geq n$,
NSPACE$(f(n)) \subseteq$ SPACE$(f^2(n))$

$\implies$ NPSPACE = PSPACE

PSPACE-completeness: Quantified Boolean Formula = valid ?
also admits log-space reducibility

L and NL

Model for *sublinear* space: read-only input tape, *work tape* gives space complexity.

log-space: constant number of pointers to input tape

PATH $= \{\langle G, s, t\rangle \mid G$ is directed graph that has an $s \rightsquigarrow t$ path $\}$ is NL-complete.

NL-coNL: Nondeterministic space complexity classes are closed under complementation (Immerman-Szelepcsényi)

$\overline{PATH} \in$ NL, guess and re-count number of reachable nodes

Traversing $\log n$-depth tree in log-space:
pointer to node at each level is $\log n \implies O(\log^2 n)$; instead:
keep constant-bit (bounded degree?) encoding of branch chosen

Complexity Hierarchies

L $\subseteq$ NL $=$ coNL $\subseteq$ **P** $\subseteq$ NP $\subseteq$ **PSPACE** $\subseteq$ EXPTIME

NL $\subsetneq$ **PSPACE** **P** $\subsetneq$ **EXPTIME**

$f : \mathbb{N} \to N$ that is at least $O(\log n)$ is *space constructible* if there is a $O(f(n))$ space TM that computes $f(n)$ from 1^n

Space Hierarchy Theorem: For *any* space constructible function $f : \mathbb{N} \to \mathbb{N}$, there exists a language that is decidable in $O(f(n))$ space but not $o(f(n))$ space.

SPACE(n^{c_1}) $\subsetneq$ SPACE(n^{c_2}) for any real $c_1, c_2 > 0$

$t : \mathbb{N} \to N$ that is at least $O(n \log n)$ is *time constructible* if $t(n)$ is computable in time $O(t(n))$ from 1^n.

Time Hierarchy Theorem: For *any* time constructible function $t : \mathbb{N} \to \mathbb{N}$, there exists a language that is decidable in $O(t(n))$ time but not in time $o(t(n)/\log t(n))$.

TIME(n^{c_1}) $\subsetneq$ TIME(n^{c_2}) for any reals $1 \le c_1 < c_2$

Circuit Complexity

Parameterized circuit families: uniformity, size-depth complexity

NC^i: decidable by a uniform family of circuits with polynomial size and $O(\log^i n)$ depth.

AC^i: like NC, but gates have unlimited fan-in (inputs).
TC^i: like AC, and also *majority* gates ("threshold circuits").

NC$^i \subseteq$ **AC**$^i \subseteq$ **TC**$^i \subseteq$ **NC**$^{i+1}$

NC $\subseteq$ **P** (generate + evaluate in polynomial time)

CIRCUIT-VALUE is **P-complete**

NC$^1 \subseteq$ **L**: evaluate in log-space (constant-bit trick per level)

NL $\subseteq$ **NC**^{2}: *PATH*/transitive closure in $\log^2 n$ depth

Branching Programs, Arithmetization

Barrington's theorem:
depth d circuit $\implies$ branching program of width 5 and length 4^d.
log-depth circuit $\implies$ poly-length program

Arithmetization:

Construct polynomials from branching programs / formulas for (dis)proving equivalence (increase probability of finding difference)

Probabilistic Complexity Classes

BPP (bounded error): accepts/rejects with error probability $\epsilon < 1/2$

Amplification lemma: can make error $2^{-p(n)}$ for any polynomial

RP (randomized poly-time): always *rejects* when it should
re-runs make acceptance error arbitrarily small
coRP: always *accepts* when it should

Clearly **RP** $\subseteq$ **BPP** (reject error is zero), likewise **coRP** $\subseteq$ **BPP**

P $\subseteq$ **RP**: no nondeterminism, always right answer

RP $\subseteq$ **NP**: NTM needs no coin, guesses correct path

BPP ? NP (no relation is known). It it believed that **P** $=$ **BPP**.

Polynomial identity testing is in coRP, unknown if in P.

Alternation

- **universal states** (AND, $\wedge$) accepts if *all* successors do
- **existential states** (OR, $\vee$) accepts if *some* successor does

Example: *MIN-FORMULA* in AP:
universally select all formulas ψ shorter than ϕ
existentially select a truth assignment, eval ψ and ϕ

Alternation and Space/Time Complexity Connections

(1) ATIME($f(n)$) $\subseteq$ SPACE($f(n)$) $\subseteq$ ATIME($f^2(n)$) for $f(n) \geq n$

(2) ASPACE($f(n)$) $=$ TIME($2^{O(f(n))}$) for $f(n) \geq \log n$

Consequences:

AL $=$ P	(2), $f(n) = \log n$
AP $=$ PSPACE	(1), $f(n) =$ poly(n)
APSPACE $=$ EXPTIME	(2), $f(n) =$ poly(n)

Polynomial Time Hierarchy

bound number of alternations between $\wedge$ and $\vee$
$\Longrightarrow$ **hierarchy** *within* AP $=$ PSPACE

Σ_i alternating TM: at most i runs of existential or universal steps, starting with existential steps.

Π_i alternating TM: at most i runs of existential or universal steps, starting with universal steps.

$\Sigma_i\mathbf{P} = \bigcup_k \Sigma_i\text{TIME}(n^k)$ $\qquad$ $\Pi_i\mathbf{P} = \bigcup_k \Pi_i\text{TIME}(n^k)$

PH $= \bigcup_i \Sigma_i$P $= \bigcup_i \Pi_i$P

P $= \Sigma_0$P $= \Pi_0$P (no nondeterminism, no alternation)

NP $= \Sigma_1$P, coNP $= \Pi_1$P.

Interactive Proofs

IP a verifier (polynomially computable function) V exists such that for every string w

1. for *some* function P, $w \in A \implies \Pr[V \leftrightarrow P \text{ accepts } w] \geq \frac{2}{3}$
2. for *any* function $\tilde{P}$, $w \notin A \implies \Pr[V \leftrightarrow P \text{ accepts } w] \leq \frac{1}{3}$

- *some* (honest) prover P can produce likely correct accept
- *no* (dishonest?) prover $\tilde{P}$ can produce likely *incorrect* accept

Can use amplification to make error probability arbitrarily small.

BPP $\subseteq$ **IP** (need no P/ ignore)

NP $\subseteq$ **IP** (never wrongly accepts, try often enough)

IP $=$ **PSPACE** (Shamir's Theorem)