CS Exit Ethiopia

Revision Outlines

Computer Science Exit Examination

1 selected course | 18 November 2026

Selected courses

01. Automata and Complexity Theory

https://csexitethiopia.com

abelasnake2003@gmail.com | Telegram: @Abechu7

Course contents

Each course begins on a new page. PDF bookmarks provide question, study-note section and revision-topic navigation.

Automata and Complexity Theory 3

Automata and Complexity Theory

Revision Outlines

Symbols, alphabets, words and length | Exam focus

- Alphabet, word and language
  - Σ: finite symbol set
  - String: finite ordered symbol sequence
  - $L \subseteq \Sigma^*$
  - Empty word $\varepsilon=\lambda$; length zero
  - $\varnothing$: no words
  - $\{\varepsilon\}$: one word
  - ε and $\varnothing \neq$ numeric 0
- Length versus word powers
  - Occurrences, not distinct symbols
  - $|caddba|=6$
  - $|caddba|^2=36$
  - $|(caddba)^2|=12$
  - Exponent placement trap
  - Choice repair requires changed expression

Concatenation, powers and reversal

- Concatenation, powers and reversal
  - cddc·ababb=cddcababb
  - Order generally matters
  - $\varepsilon x=x\varepsilon=x$
  - $x^0=\varepsilon$; x^n: n copies
  - $caddba^R=abddac$
  - $(xy)^R=y^R x^R$
  - $(x^R)^R=x$
- Language concatenation and powers
  - $LM=\{xy:x\in L,y\in M\}$
  - $L^0=\{\varepsilon\}$; $L^{n+1}=L^n L$

- {e,fgh}{c,d}={ec,ed,fghc,fghd}
- fghe absent
- $L^0=\varepsilon$: missing-braces type error
- Case-sensitive symbols
- Uppercase/lowercase transcription requires declared repair

Language Operations and Empty-Word Cases | Exam focus

• Alphabet closures and word counts
- Σ^n: exactly n symbols
- $|\Sigma|=k \Rightarrow |\Sigma^n|=k^n$
- $\Sigma^0=\{\varepsilon\}$
- $\Sigma^*=\bigcup n\geq 0\Sigma^n$
- $\Sigma^+=\bigcup n\geq 1\Sigma^n$
- Nonempty alphabet: $\varepsilon\notin\Sigma^+$
- $\Sigma^*=\Sigma^+\cup\{\varepsilon\}$
- $\{a,b\}^2=\{aa,ab,ba,bb\}$

• Language operations and empty cases
- $L^*=\bigcup n\geq 0L^n$; $L^+=LL^*$
- $\varepsilon\in L \Rightarrow \varepsilon\in L^+$
- Alphabet positive-closure distinction
- $\varnothing^*=\{\varepsilon\}$; $\varnothing^+=\varnothing$
- $\{\varepsilon\}^*=\{\varepsilon\}$
- Union: either set
- Intersection: same word in both
- Intersection ≠ concatenation
- Complement Σ^*-L: fixed alphabet

Grammars generate; automata recognize | Exam focus

• Grammar versus recognizer
- $G=(V,\Sigma,P,S)$
- Finite variables; disjoint terminals
- Productions; start $S\in V$
- Left-side replacement

- L(G)={w∈Σ*:S⇒*w}
- Grammar: language generation
- Automaton: word acceptance/language recognition
- Automata recognize languages, not grammars

• Language equivalence and notation
- Equivalent grammars: equal generated languages
- Rule counts/names irrelevant
- Common word insufficient
- Equal alphabets insufficient
- One-way inclusion insufficient
- RE denotes language, not word set
- Formal examples: precisely specified sets
- Natural-language example trap

Chomsky Hierarchy and Epsilon Conventions | Exam focus

• Chomsky model correspondences
- Type 3 REG ↔ finite automata
- Type 2 CFG ↔ NPDA
- Type 1 CSL ↔ nondeterministic LBA
- Context-sensitive/noncontracting grammars
- Type 0 unrestricted ↔ Turing recognizers
- Unrestricted α→β: variable in α
- Meaningless arbitrary rewrite trap

• Expressiveness hierarchy and ε exception
- REG⊊CFL⊊CSL⊊DECIDABLE⊊RE
- Optional S→ε; S absent from RHS
- Strict noncontracting rules cannot generate ε
- Stronger models include weaker languages
- Capable-family questions: potentially nonunique
- Least sufficient versus exact equivalent family
- Expressiveness ≠ runtime ranking

DFA definition and acceptance | Exam focus

- DFA formal definition
  - $M=(Q,\Sigma,\delta,q_0,F)$
  - Finite nonempty Q; $q_0 \in Q$
  - $F \subseteq Q$, possibly empty
  - Total $\delta: Q \times \Sigma \to Q$
  - One successor per state/symbol pair
  - One-transition-machine misconception
  - Empty F recognizes $\varnothing$
  - State names ≠ acceptance status
- DFA acceptance and memory
  - Entire input consumed
  - Ending state in F
  - Earlier accepting visit insufficient
  - ε accepted iff $q_0 \in F$
  - Rejecting sink: all-symbol self-loops
  - Sink still finishes finite input
  - Finite state-encoded memory
  - Finite memory ≠ no memory

Finite-State Recognizer Design and Traces | Exam focus

- Suffix-invariant DFA design
  - Binary ending 01
  - q_0: no useful suffix
  - q_1: suffix 0; q_2: suffix 01
  - Only q_2 accepts
  - (0,1): $q_0 \to (q_1, q_0)$
  - (0,1): $q_1 \to (q_1, q_2)$
  - (0,1): $q_2 \to (q_1, q_0)$
  - 10101: $q_0, q_0, q_1, q_2, q_1, q_2$
  - Final q_2: accepted
- Length-at-least-two DFA
  - States: 0, 1, ≥2 symbols

- Every symbol advances count stage
- Third state: accepting loop
- q_2 naming arbitrary
- Missing diagram/meanings: no numbered answer
- Test ε and shortest boundary words
- Test overlaps
- Sample agreement ≠ proof

Minimum States for Prefixes and Suffixes | Exam focus

• Fixed-suffix DFA minimum
  - Valid word length n: n+1 states
  - Longest suffix matching pattern prefix
  - Overlapping-prefix fallback transitions
  - Distinguishable prefixes via continuations
  - Lower-bound proof
  - Unnecessary rejecting-sink trap
• Fixed-prefix DFA minimum and alphabet
  - n matched-prefix stages
  - Accepting continuation and rejecting sink
  - n≥1, mismatches possible: n+2
  - Unary alphabet: potentially unreachable sink
  - Pattern symbol outside Σ: empty language
  - Empty-language minimum: one rejecting state
  - Starting abc over {a,b}: one state
  - Five requires alphabet containing a,b,c

NFA and epsilon transitions | Exam focus

• NFA and ε-NFA
  - $\delta: Q \times \Sigma \to P(Q)$
  - Zero, one, many successors
  - ε-NFA domain: $Q \times (\Sigma \cup \{\varepsilon\})$
  - ε moves consume nothing
  - NFA convention varies

- Existential finite accepting path
- Complete word consumed
- ε moves before/after input symbols
- Other paths may get stuck

• NFA versus DFA power
- Both recognize exactly REG
- Nondeterminism: potentially smaller description
- DFA successor as singleton set
- ε ≠ input character
- ε ≠ permission for unread input

Subset construction with epsilon closure | Exam focus

• Subset construction with ε-closure
- ε-closure(T): T plus ε-reachable states
- Start: ε-closure($\{q_0\}$)
- Symbol step: ε-closure($\bigcup_{q \in T} \delta(q,a)$)
- Accepting subset intersects F
- Reachable subsets only
- Empty subset: rejecting sink if reached
- n-state NFA: at most 2^n states
- Exactly-2^n trap

• Subset-construction example
- $\Sigma=\{a\}$; start p; final r
- $\delta(p,a)=\{p,r\}$; $\delta(r,a)=\varnothing$
- Start $\{p\}$
- $a \rightarrow \{p,r\}$; further a self-loop
- Accepted language a^+
- Added ε edge $p \rightarrow r$
- Start already accepting; ε accepted

Equivalence, minimization and diagram checks | Exam focus

• DFA minimization
- Discard unreachable states

- Initial accepting/nonaccepting partition
- Split by transition target blocks
- Repeat until stable
- Stable equivalence classes merged
- Equivalent: every continuation agrees
- Current acceptance alone insufficient

• Equivalence and diagram checks
- Reachable product-state pairs
- Exactly one accepting: distinguishing word
- No disagreement: language equality
- Complete partial diagram with sink
- RE test: ε and shortest cycles
- Accepting start plus ab return
- Equivalent expression must include ε,ab

Regular-expression notation and precedence

• Classical RE syntax and precedence
- ∅; ε; symbols; union; concatenation; star
- Union: A|B or A+B
- Postfix r^+: one-or-more
- Distinct uses of +
- Parenthesize conflicting conventions
- Star > concatenation > union
- r?=r|ε
- r^+=rr*

• RE interpretation examples
- (a|b)*a: ending a
- 0*10*: exactly one 1
- a*(b|c): a's then one b/c
- aab accepted; aaa,aaabc rejected
- (0|ε)(0|1)*: all binary words
- ε alternative permits any suffix
- Programming backreferences may exceed REG

- Wildcard dot: additional notation

Regular Expressions and Representation Conversions | Exam focus

- Build regular expressions
  - At least one a: (a|b)*a(a|b)*
  - Exactly one a: b*ab*
  - At least two zeros: (0|1)*0(0|1)*0(0|1)*
  - Nonadjacent zeros allowed
  - 010 distinguishes from substring 00
  - S→aS|ε generates a*
- RE-automaton conversions
  - RE→ε-NFA fragments
  - Union: parallel alternatives
  - Concatenation: connected fragment ends
  - Star: ε bypass and repetition
  - Then subset construction
  - Automaton→RE: state elimination
  - $r_{ij} \mid r_{ik}(r_{kk})^*r_{kj}$
  - New start/final nodes when required
  - Whole-language preservation

Regular grammars and closure constructions | Exam focus

- Regular grammar orientations
  - Right-linear: A→aB, A→a, permitted ε
  - Left-linear: A→Ba
  - Consistent orientation: REG
  - Arbitrary mixing may be nonregular
  - S→aA|ε, A→Sb: a^nb^n
  - A→aBA: CFG, not regular form
- Regular constructions and closure
  - Variables → NFA states
  - A→aB: a edge

- A→a: fresh final state
- A→ε: accepting A
- Union; concatenation; star; reversal
- Intersection; difference; complement
- Reversal: reverse edges, swap start/finals
- ε-NFA handles reversed starts
- Complement: complete DFA, swap acceptance
- Product states: intersection/difference

Regular pumping lemma: use the quantifiers | Exam focus

• Regular pumping lemma
  - $\exists p \geq 1$; $\forall s \in L$, $|s| \geq p$
  - $s=xyz$; $|xy| \leq p$; $|y| \geq 1$
  - $\forall i \geq 0$: $xy^iz \in L$
  - Nonregular proof: arbitrary p, chosen witness
  - Every allowed split broken
  - One bad split insufficient
  - Necessary condition ≠ regularity proof
• Pumping a^nb^n and infinitude
  - Witness a^pb^p
  - Every split: $y=a^k$, $k \geq 1$
  - i=0: fewer a's than b's
  - Contradiction excludes every DFA
  - n-state DFA: long path repeats state
  - Accepted cycle: indefinite repetition
  - Infinite iff accepted word length ≥n

CFG derivations, sentences and parse trees | Exam focus

• CFG productions and sentential forms
  - $A \to \alpha$; $A \in V$; $\alpha \in (V \cup \Sigma)^*$
  - One variable on left
  - Weaker set-membership options also true
  - Sentential form: any S-derived string

- Includes terminal sentences and S

• Derivations and parse trees
  - Leftmost: leftmost variable replacement
  - Rightmost: rightmost variable replacement
  - Not opposite input-scanning directions
  - Root S; expanded internal variables
  - Terminal leaves; optional ε leaves
  - Left-to-right yield
  - Tree: derivation structure
  - Derivation-graph path: different object

Design and validate context-free grammars | Exam focus

• Matched-structure CFG design
  - Recursive rule plus stopping base
  - $S \to aSb \mid \varepsilon$: $a^n b^n$, $n \geq 0$
  - n recursions: n matched pairs
  - $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$
  - $n \geq 1$: base ab replaces ε
• CFG construction and proof
  - $a^n b^m$, $n > m \geq 0$
  - $S \to aSb \mid A$; $A \to aA \mid a$
  - S: matched pairs; A: extra a's
  - Balanced parentheses: $S \to (S)S \mid \varepsilon$
  - Soundness: preserved intended property
  - Completeness: decompose desired words
  - Several examples ≠ language equality

Ambiguity and precedence | Exam focus

• CFG ambiguity
  - One word, two distinct parse trees
  - Equivalent: two distinct leftmost derivations
  - Same-tree expansion orders insufficient
  - $E \to E+E \mid E*E \mid id$

- id+id*id: (id+id)*id versus id+(id*id)

• Precedence and ambiguity distinctions

- E→E+T|T; T→T*F|F; F→id|(E)
- Multiplication before addition
- Left recursion: left association
- Uniqueness: last-operator split and induction
- S→SS|ε: ambiguous ε
- S→A|B; both branches derive ab
- A→ab; B→abB|ε
- Ambiguous grammar ≠ inherently ambiguous language
- Alternative grammar may remove ambiguity

Simplification, CNF and membership | Exam focus

• Grammar simplification

- Null A→ε; unit A→B
- Mark nullable variables
- Omit nullable occurrences through alternatives
- Fresh start preserves ε
- Unit-chain reachability copies nonunit productions
- Remove nongenerating, then unreachable
- Nullable deletion: nonempty-word loss trap

• CNF and CYK membership

- A→BC or A→a
- Optional isolated $S_0 \to \varepsilon$
- Long-rule terminals replaced by variables
- Binary variable-sequence splitting
- $A \to aBC \Rightarrow A \to XC_1$, $X \to a$, $C_1 \to BC$
- CYK: substring variable sets
- Combine splits using A→BC
- Fixed grammar: $O(n^3)$
- n>0: n terminal, n−1 binary rules

Greibach form and left recursion | Exam focus

- Greibach normal form
  - $A \to aA_1 \ldots A_k$, $k \geq 0$
  - Leading terminal, then only variables
  - ε: isolated fresh start
  - Fresh start absent from right sides
  - Length $n>0$: n leftmost rule applications
  - Each step adds one prefix terminal
- GNF conversion choices
  - Remove ε, unit, useless rules
  - Order variables; substitute earlier-leading rules
  - Remove left recursion
  - Substitute until terminal-leading
  - CNF-first: optional algorithm choice
  - $A \to A\alpha \mid \beta \Rightarrow A \to \beta R$, $R \to \alpha R \mid \varepsilon$
  - Further ε normalization
  - No direct or indirect left recursion
  - First-step question requires named algorithm

PDA configuration and acceptance | Exam focus

- PDA structure and transitions
  - $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$
  - Finite control plus stack
  - State,input/ε,top → new-state,replacement pairs
  - Stack-top orientation before traces
  - X→X: unchanged stack
  - ε input: no consumption
  - X→ε: pop
  - Distinct ε roles
- PDA configurations and acceptance
  - State; unread input; stack
  - Final-state mode: consumed input, final state
  - Empty-stack mode: consumed input, empty stack

- NPDA modes: same language family
- Conversion, not same unmodified machine
- DPDA acceptance conventions need care

PDA Construction and Determinism | Exam focus

- PDA for $a^n b^n$
  - Push X per a
  - Switch to b phase
  - Pop X per b
  - End input at bottom marker
  - Reject a after b phase
  - Reject b without X
  - Explicit ε handling for n=0
  - Stack counting ≠ random-access input storage
- Mirror versus copy and determinism
  - ww^R: push, guess midpoint, match pops
  - Binary unmarked midpoint: CFL, not DCFL
  - wcw^R: visible switch, DPDA sufficient
  - Unary even palindromes: even length, REG
  - ww: same-forward-order copy
  - Binary copy language ≠ palindrome language
  - Single-stack comparison limitation

CFL closure and counterexamples | Exam focus

- CFL closure with regular intersection
  - Union; concatenation; star; reversal
  - CFL∩REG: context-free
  - PDA×DFA control; original stack
  - Consumed symbol updates both states
  - ε move leaves DFA unchanged
  - Not necessarily regular
  - $a^n b^n \cap a^* b^* = a^n b^n$
- CFL nonclosure boundaries

- Arbitrary intersection not closed
- $a^n b^n c^m$ and $a^m b^n c^n$: CFL
- Intersection $a^n b^n c^n$: non-CFL
- Complement/arbitrary difference not closed
- CFL−REG: CFL
- Intersect with regular complement
- Failed proposed PDA ≠ impossibility proof

CFL pumping proof and copy versus mirror | Exam focus

- CFL pumping lemma
  - CFL assumption; some pumping length p
  - Every sufficiently long $s \in L$
  - $s=uvxyz$; $|vxy| \le p$; $|vy| \ge 1$
  - $\forall i \ge 0$: $uv^i xy^i z \in L$
  - v and y pumped together
  - Either empty, not both
  - Non-CFL proof: arbitrary pumping length
  - Every allowed decomposition defeated
- Three-count and copy-language boundaries
  - Witness $a^p b^p c^p$
  - Window cannot contain a and c
  - At most two blocks affected
  - At least one count changes
  - i=0 or 2
  - Third count fixed or order broken
  - $a^n b^n c^n$: non-CFL
  - Binary ww: non-CFL
  - ww^R grammar: $S \to aSa \mid bSb \mid \varepsilon$
  - Calling ww palindrome: contradictory

Context-sensitive rules and linear bounded automata | Exam focus

- Context-sensitive and noncontracting rules

- αAβ→αγβ; γ≠ε
- Context-preserving variable replacement
- Complete-side length nondecrease
- Isolated ε exception
- Unrelated |β|≥|α| comparison trap

• LBA and CSL decidability
- Length-n input: O(n) tape cells
- CSL ↔ nondeterministic LBA
- $a^nb^nc^n$: verify a*b*c* order
- Repeated one-a,b,c marking
- Missing/excess symbols rejected
- Marks confined to input region
- m cells: ≤|Q|m|Γ|^m configurations
- Finite-graph acceptance reachability
- Looping branches allowed
- Every CSL decidable

Language Classification and Boundaries | Exam focus

• Language-family classification
- a*b* and a^nb: REG
- a^nb^n: CFL, nonregular
- Binary ww^R: CFL
- Binary ww: non-CFL
- Both mirror/copy: decidable
- $a^nb^nc^n$: CSL, non-CFL, decidable
- Finite languages: regular
- Stronger models recognize lower-family languages

• Unary prime-length boundary
- {a^p:p prime}: nonregular
- Choose prime q>pumping length p
- Pump piece length k≥1
- Pump exponent q+1
- Length q(k+1): composite

- Unary CFL ⇒ regular
- Prime-length language: non-CFL
- Bounded-workspace divisor test: CSL
- Listed examples: none nonrecursive

TM components, transitions and configurations | Exam focus

- Turing machine components
  - $M=(Q,\Sigma,\Gamma,\delta,q_0,q\text{accept},q\text{reject})$
  - Finite state/tape alphabets
  - $\Sigma\subseteq\Gamma$; blank $\square\in\Gamma-\Sigma$
  - Module blank$\notin\Gamma$: wrong
  - Finite control; read/write head; unbounded tape
  - Input: initial tape content
  - Separate input organ/stack unnecessary
- TM transitions and instantaneous descriptions
  - $\delta(q,a)=(p,b,L)$
  - Read a, write b, left, state p
  - Halting states: no moves
  - ID u q av: scans a
  - Left: $u=u'c \Rightarrow u'\ p\ cbv$
  - Right: ub p v
  - Unused tape: show blank
  - Diagram-specific state/transition conventions

TM Design, Variants and Limits | Exam focus

- TM decision design and halting
  - $a^nb^nc^n$: mark triples, reset scans
  - Verify no unmatched symbols
  - Progress: additional marks each round
  - Finite rounds
  - ww: reject odd length
  - Mark/counter midpoint
  - Corresponding-half symbol comparison

- Writable tape beyond single-stack access
- Acceptance test and all-input halting proof

• TM variants and Church-Turing thesis

- Off-line: read-only input, read/write work
- Multitape; multihead; two-way-infinite tape
- Nondeterministic TM: same recognition power
- Different simulation costs
- Two unbounded stacks simulate tape
- Turing completeness: general TM simulation
- No polynomial-speed implication
- Finite runs need finite cells
- Effective procedures: Church-Turing thesis
- Not proved runtime theorem

Deciders, recognizers and complements | Exam focus

• Deciders versus recognizers

- Decider: halts on every input
- Recursive = decidable
- Recognizer: members halt-accept
- Nonmembers: halt-reject or loop
- RE = recognizable = semidecidable
- Loop ≠ rejecting halt
- Undecidable may remain recognizable

• Complements and computable functions

- DECIDABLE $\subsetneq$ RE
- L and complement RE $\Rightarrow$ decidable
- Dovetail recognizers until acceptance
- Decider complement: swap outcomes
- Recognizer final-state flipping fails on loops
- Total computable functions
- Recursive function ≠ recursive self-calls
- Partial functions undefined on nonhalting computations

Halting, diagonalization and Rice's theorem | Exam focus

- HALT and diagonalization
  - HALT={⟨M,w⟩:M halts on w}
  - Recognizable: simulate, accept any halt
  - Assume halting decider H
  - D(x): loop if H(x,x) predicts halt
  - Otherwise halt
  - D on own description: contradiction
  - HALT: RE, undecidable
  - Complement HALT not RE
  - Both recognizable would yield decider
- Rice's theorem and limits
  - Nontrivial recognized-language semantic property
  - Some RE languages satisfy, others not
  - Empty language; contains ε; ≥3 words
  - Undecidable language properties
  - State count: syntactic
  - Symbol-writing: machine behavior
  - Nonlanguage properties require separate proof
  - Undecidable: no general decider
  - Currently unknown algorithm ≠ undecidability

Decidable and Undecidable Questions | Exam focus

- Decidable automata/grammar questions
  - DFA membership: simulation
  - NFA membership: reachable sets
  - FA emptiness: reachable final state
  - DFA equivalence: product acceptance disagreement
  - CFG membership: parsing/CYK
  - CFG emptiness: generating-variable marking
  - Halting procedures
  - Arbitrary TM membership distinction
- Undecidable questions and syntax scope

- TM acceptance: RE-complete, undecidable
- HALT: undecidable
- General CFG ambiguity/equivalence: undecidable
- Fixed CFG-word membership: decidable
- Two trees establish particular ambiguity
- No universal ambiguity decider follows
- FORTRAN: syntax versus semantics versus computation
- Language name alone insufficient

Computable reductions for undecidability | Exam focus

- Computable many-one reduction
  - $A \leq_m B$: total computable f
  - $x \in A$ iff $f(x) \in B$
  - B decider plus f decides A
  - Known-undecidable A → target B
  - $B \leq_m A$ alone: no B-hardness proof
- HALT$\leq_m$A_TM construction
  - Given $\langle M,w \rangle$, construct N
  - N ignores input, simulates M(w)
  - Accept on any M halt
  - Rejecting halts included
  - $\langle N,\varepsilon \rangle$ accepted iff M(w) halts
  - Finite-description construction terminates
  - Internal simulation may loop
  - Both yes/no directions
  - Computable mapping, not resemblance

Resources and Input Encoding | Exam focus

- Resource measures and asymptotic bounds
  - Time: computation steps
  - Space: used cells/memory
  - Read-only input inclusion convention
  - Deterministic decider: length-n maximum

- Bounded NTM: every branch bounded
- Big-O: asymptotic upper bound
- Worst/average/best-case functions all eligible
- Θ: matching lower bound

• Growth and encoding size

- $1 < \log n < n$
- $n < n \log n < n^2$
- $n^2 < 2^n < n!$
- Fixed polynomial n^k versus exponential 2^n
- Binary N: Θ(log N) bits
- O(N) not polynomial in binary length
- Pseudopolynomial: numeric magnitude dependence
- Huge polynomial degree: potentially impractical

P, NP and polynomial verification | Exam focus

• P and NP definitions

- Classes of decision languages
- P: deterministic polynomial-time decider
- NP: nondeterministic polynomial-time decider
- Polynomial verifier and polynomial-size certificate
- Yes-input: some accepting certificate
- No-input: no accepting certificate
- NP ≠ non-polynomial

• CLIQUE verification and $P \subseteq NP$

- Certificate: k vertices
- Check distinctness and every pair-edge
- Polynomial in graph encoding
- P verifier ignores certificate
- P=NP: open
- NP ≠ definitely hard/outside P
- Finding/verifying intuition, not exponential-search necessity

Polynomial Reductions and Consequences | Exam focus

- Polynomial reduction consequences
  - $A \leq_p B$: polynomial f
  - $x \in A$ iff $f(x) \in B$
  - Solving B solves A
  - B at least A-hard
  - $B \in P \Rightarrow A \in P$
  - A NP-hard $\Rightarrow$ B NP-hard
  - $A \in P \nRightarrow B \in P$
  - Same-class inference trap
  - Polynomial reductions compose
- Hardness direction and mapping bounds
  - Known-hard Y → new target X
  - X→Y: reversed-proof trap
  - Polynomial mapping time and instance size
  - Undecidability: computable reduction sufficient
  - NP-completeness: arbitrary computability insufficient
  - Explicit reduction and resource bound

NP-Hardness, Completeness and Decision Formulations | Exam focus

- NP-hard versus NP-complete
  - NP-hard: every NP language $\leq_p X$
  - NP-hard need not lie in NP
  - NP-hard may be undecidable
  - NP-complete: NP membership plus NP-hardness
  - NP-complete lies inside NP
  - Finite exhaustive certificates: exponential decision
  - Multiple true source options
  - Exponential necessity unproved
- Canonical complexity examples
  - SAT: satisfying Boolean assignment
  - Cook-Levin: SAT NP-complete

- CNF-SAT and 3-SAT: NP-complete
- TSP decision: tour cost ≤B
- Standard finite encoding: NP-complete
- Shortest-tour optimization: NP-hard
- NP-complete requires decision formulation
- Fixed 2-colorability: P
- Fixed k≥3 colorability: NP-complete

3-SAT to CLIQUE Hardness Proof | Exam focus

- 3-SAT→CLIQUE construction
  - CLIQUE∈NP: pairwise-edge certificate check
  - m clauses
  - Vertex per literal occurrence
  - Edges: different clauses, noncomplementary literals
  - Target clique size m
  - Polynomial construction time and space
- 3-SAT→CLIQUE equivalence and correction
  - Satisfying assignment: true literal per clause
  - Consistent literals form clique
  - m-clique: one occurrence per clause
  - No complementary chosen literals
  - Assign selected literals true
  - Extend remaining variables arbitrarily
  - Both implications
  - Source formula: $(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge \neg x_3$
  - (true,true,false) satisfies all clauses
  - ¬false=true: source correction

PSPACE, quantifiers and space reuse | Exam focus

- PSPACE and memory reuse
  - PSPACE=$\cup_k$SPACE(n^k)
  - Deterministic polynomial workspace
  - NP⊆PSPACE: enumerate/check witnesses sequentially

- Exponential candidates, reused memory
- Savitch: NSPACE(s)⊆DSPACE(s^2)
- Appropriate s≥log n
- NPSPACE=PSPACE
- No P=NP implication

• TQBF and quantifier evaluation
- Fully quantified Boolean truth
- TQBF: PSPACE-complete
- ∃x: either branch
- ∀x: both branches
- DFS assignments and recursion state
- Polynomial space, potentially exponential time
- ∀x∃y(x↔y): true
- ∃y∀x(x↔y): false
- Membership plus every-PSPACE-language hardness
- Stated reduction qualifier

Known Hierarchy and Open Separations | Exam focus

• Complexity containment hierarchy
- L⊆NL⊆P⊆NP⊆PSPACE⊆EXPTIME⊆EXPSPACE
- L/NL: logarithmic workspace
- EXPTIME=∪kTIME(2^(n^k))
- Upper-bound class includes P
- Membership ≠ exponential necessity
- Hierarchy: P⊊EXPTIME
- Hierarchy: PSPACE⊊EXPSPACE
- Not all adjacent separations established

• Open separations and collapse limits
- P versus NP: open
- NP versus PSPACE: open
- PSPACE versus EXPTIME: open
- coNP: complements of NP languages
- P⊆NP∩coNP

- NP=coNP: open
- NP-complete in P ⇒ P=NP
- No all-decidable polynomial collapse
- Undecidable remains undecidable
- Cryptography/easiness claims require class and encoding

Unusual Bank Grammar Derivations

- Unusual right-linear grammar invariant
  - S→11S|0A|1A; A→0|1|0S
  - At most one rightmost variable
  - Terminal word still sentential form
  - S⇒11S⇒1111S⇒111111S⇒1111110A⇒11111101
  - Return-to-S pairs: 11,00,10
  - Prefix invariant over length guessing
- Unusual grammar membership repair
  - A→aB|bA|bBa; B→bB|cA|b
  - Start variable required
  - Assume start A
  - Remove required a/b prefix
  - Remove b…a wrapper
  - B: match b, or recurse bB/cA
  - Shorter residual guarantees termination
  - All four offered words fail
  - Marked choice contradicted; no option works

Source Defects and Missing Information

- Formula and diagram prerequisites
  - Exact readable language and alphabet
  - Arbitrary state labels
  - Transition diagram and initial state
  - Accepting markings or state meanings
  - Theory alone cannot identify labels
- Specific source defects

- Accepting start and ab-return paths
- Required ε and ab absent together
- Trace strings from the initial state
- Forward copy ww versus palindrome ww^R
- Capable/contains: least-family or exact-class qualifier
- Most difficult: reduction qualifier
- GNF first step: named algorithm
- Overlapping options: no invented unique key