==
FORMAL LANGUAGES EXAM - COMPLETE CHEAT SHEET WITH EXAMPLES
Exam: Today 2:00 PM | 90 minutes | 4 problems | Need 49/70 to pass
==

WRITE THIS ON SCRATCH PAPER FIRST (2 minutes):

TIME BUDGET:
2:10 PM - Problem 3 done (Countability)
2:35 PM - Problem 4 done (Decidability)
2:55 PM - Problem 2 done (Binary Mod)
3:10 PM - Problem 1 done (TM Design)
3:30 PM - Submit

DECIDABILITY LOOKUP TABLE:
DFA problems (E_DFA, ALL_DFA, EQ_DFA): YES/YES ✅✅
CFG membership (A_CFG, E_CFG): YES/YES ✅✅
CFG equality (EQ_CFG): NO/NO ❌❌
TM problems (A_TM, HALT_TM, E_TM): NO/mixed ❌⚠️

COUNTABILITY RULES:
Finite, ℕ, ℤ, ℚ, pairs → COUNTABLE ✓
ℝ, intervals, P(ℕ) → UNCOUNTABLE ✗
Strategy: Enumerate by sum or length

==
PROBLEM 1: TM DESIGN FOR FINITE LANGUAGE (20 points) - DO LAST!
==

STEP 1: WRITE EXAMPLES FIRST! (1 minute)

Example Problem: "Design TM that accepts A = {0, 00, 010}"

Write test cases:
✓ "0" → ACCEPT
✓ "00" → ACCEPT
✓ "010" → ACCEPT
✗ "1" → REJECT
✗ "000" → REJECT
✗ "0100" → REJECT

STEP 2: IDENTIFY PROBLEM TYPE (30 seconds)

TYPE A: Finite Specific Strings (like {0, 00, 010})
→ States = one per string + intermediate steps

TYPE B: Pattern at End (like "ends with 101")
  → States = remember last N symbols

TYPE C: Contains Pattern (like "contains 11")
  → Once found, stay in accept zone

STEP 3: DESIGN STATES (2 minutes)

Example for A = {0, 00, 010}:

q_0 = START (haven't read anything)
q_1 = SEEN "0" (could be "0", "00", or start of "010")
q_2 = SEEN "00" or "01"
q_3 = SEEN "010"
qaccept, qreject

Rule: States track "how far into a valid string are we?"

STEP 4: WRITE TRANSITIONS (5 minutes)

Go state by state, symbol by symbol:

q_0 (START):
  0 → 0,R,q_1 (could be any of our strings)
  1 → 1,R,qreject (none start with 1)
  ⊔ → ⊔,R,qreject (empty not in language)

q_1 (SEEN "0"):
  0 → 0,R,q_2 (could be "00" or start of "010")
  1 → 1,R,q_2 (could be start of "010")
  ⊔ → ⊔,R,qaccept (string is "0" ✓)

q_2 (SEEN "00" or "01"):
  0 → 0,R,q_3 (if was "01", now "010" ✓)
  1 → 1,R,qreject (too long)
  ⊔ → ⊔,R,qaccept (if was "00" ✓)

q_3 (SEEN "010"):
  0 or 1 → qreject (too long)
  ⊔ → ⊔,R,qaccept (string is "010" ✓)

STEP 5: TEST ONE EXAMPLE (2 minutes)

Test "010":

```
[0][1][0][⊔]
 ↑ q₀ → see 0 → q₁

[0][1][0][⊔]
    ↑ q₁ → see 1 → q₂
```

[0][1][0][␣]
↑ q_2 → see 0 → q_3

[0][1][0][␣]
↑ q_3 → see ␣ → qaccept ✓

PATTERN AT END EXAMPLE: "Ends with 101"

States track last 3 symbols:

q_0 = no pattern progress
q_1 = last symbol "1"
q_2 = last 2 symbols "10"
q_3 = last 3 symbols "101" (WINNING!)

Key transitions:
q_2 (have "10"):
1 → 1,R,q_3 (PATTERN COMPLETE!)
0 → 0,R,q_0 (broke pattern)

q_3 (have "101"):
1 → 1,R,q_1 (keep reading, start new pattern)
0 → 0,R,q_0 (broke pattern)
␣ → ␣,R,qaccept (STRING ENDS WITH "101"!)

TIME: ~15 minutes | TARGET: 14/20 points

==
PROBLEM 2: BINARY MODIFICATION (15 points) - DO THIRD!
==

STEP 1: WRITE EXAMPLES (1 minute)

Example Problem: "Design TM for w+1 (increment)"

"1" + 1 = "10" (1→2) ✓
"10" + 1 = "11" (2→3) ✓
"11" + 1 = "100" (3→4) ← CARRYING!
"111" + 1 = "1000" (7→8) ← OVERFLOW!

STEP 2: WRITE ENGLISH ALGORITHM FIRST! (5 minutes - 5 FREE POINTS!)

ALGORITHM:

1. Scan right across tape until blank (␣), marking end of number
2. Move left to rightmost digit
3. Check rightmost digit:
   - If 0: change to 1, ACCEPT (done!)
   - If 1: change to 0 (carry), move left
4. Continue carrying:

```
  - For each 1: change to 0, continue left
  - When find 0: change to 1, ACCEPT
  - If reach blank: write 1 (overflow), ACCEPT

EXAMPLES:
"10" → rightmost 0 → change to 1 → "11" ✓
"11" → rightmost 1 → change to 0, carry left → find 1 → change to 0,
      carry left → hit blank → write 1 → "100" ✓

STEP 3: IDENTIFY STATES (1 minute)
----------------------------------
Binary operations have 3 PHASES:

PHASE 1 - POSITIONING: qstart (scan right)
PHASE 2 - MODIFICATION: qinc (at rightmost), qcarry (carrying left)
PHASE 3 - DECISION: qaccept

4 states total for increment

STEP 4: WRITE TRANSITIONS (8 minutes)
-------------------------------------
qstart (SCAN RIGHT):
  0 → 0,R,qstart  (keep scanning)
  1 → 1,R,qstart  (keep scanning)
  ⊔ → ⊔,L,qinc  (found end! move back)

qinc (AT RIGHTMOST DIGIT):
  0 → 1,L,qaccept  (easy case! 0→1, done)
  1 → 0,L,qcarry  (need carry! 1→0, go left)
  ⊔ → 1,L,qaccept  (empty string → "1")

qcarry (CARRYING LEFT):
  1 → 0,L,qcarry  (still carrying, keep going)
  0 → 1,L,qaccept  (found 0! change to 1, done)
  ⊔ → 1,L,qaccept  (overflow! write new 1)

qaccept: HALT ◎

STEP 5: TEST "11" (2 minutes)
-----------------------------
Initial: [1][1][⊔]
         ↑ qstart

[1][1][⊔]
   ↑ qstart (moved right)

[1][1][⊔]
      ↑ qstart → see ⊔ → qinc, move L

[1][1][⊔]
   ↑ qinc → see 1 → write 0, qcarry, move L
```

```
[1][0][⊔]
 ↑ qcarry → see 1 → write 0, qcarry, move L

[0][0][⊔]
↑ qcarry → see ⊔ → write 1, qaccept
```

Result: [1][0][0] = "100" = 4 ✓ CORRECT!

DECREMENT EXAMPLE (w-1):

States: qstart, qdec, qborrow, qaccept, qreject

```
qdec (AT RIGHTMOST):
  1 → 0,L,qaccept  (easy! 1→0, done)
  0 → 1,L,qborrow  (need borrow! 0→1, go left)
  ⊔ → ⊔,R,qreject  (empty = can't decrement)

qborrow (BORROWING LEFT):
  0 → 0,L,qborrow  (skip 0s, keep looking)
  1 → 0,L,qaccept  (found 1! change to 0, done)
  ⊔ → ⊔,R,qreject  (reached start = was "0")
```

TIME: ~20 minutes | TARGET: 13/15 points (ENGLISH ALGORITHM = 5 FREE!)

==
PROBLEM 3: COUNTABILITY PROOF (10 points) - DO FIRST!
==

QUICK DECISION (30 seconds):

Is it...

```
├── Finite? → COUNTABLE ✓
├── ℕ, ℤ, ℚ? → COUNTABLE ✓
├── Pairs/triples? → COUNTABLE ✓
├── Strings over finite alphabet? → COUNTABLE ✓
├── ℝ, intervals [0,1]? → UNCOUNTABLE ✗
└── P(ℕ)? → UNCOUNTABLE ✗
```

EXAMPLE 1: Is ℕ×ℕ countable? (7 minutes)

ANSWER: YES, ℕ×ℕ is countable. ✅

JUSTIFICATION:

We can enumerate all pairs systematically by their sum (i+j):

Sum 0: (0,0)
Sum 1: (0,1), (1,0)
Sum 2: (0,2), (1,1), (2,0)
Sum 3: (0,3), (1,2), (2,1), (3,0)
Sum 4: (0,4), (1,3), (2,2), (3,1), (4,0)
...

This enumeration is systematic and covers every pair exactly once:
- Any pair (i,j) appears at sum level i+j
- Each sum level has finitely many pairs
- We can list them in order: sum 0, sum 1, sum 2, ...

Therefore, ℕ×ℕ is countable. ✓

EXAMPLE 2: Is [0,1] countable? (7 minutes)

ANSWER: NO, [0,1] is NOT countable. ❌

JUSTIFICATION:

By Cantor's diagonalization argument, the interval [0,1] is uncountable.

Proof sketch:
1. Assume [0,1] is countable (for contradiction)
2. List all real numbers in [0,1]: r_1, r_2, r_3, ...
3. Construct new number d by changing nth decimal of r_n
4. d is in [0,1] but NOT in our list! Contradiction.
5. Therefore, [0,1] is uncountable. ✗

Since [0,1] ⊆ ℝ and [0,1] is uncountable, ℝ is also uncountable.

EXAMPLE 3: Is {0,1}* (all binary strings) countable? (7 minutes)

ANSWER: YES, {0,1}* is countable. ✅

JUSTIFICATION:

We can enumerate all binary strings by length:

Length 0: ε
Length 1: 0, 1
Length 2: 00, 01, 10, 11
Length 3: 000, 001, 010, 011, 100, 101, 110, 111
...

This enumeration covers all strings:
- Every string has finite length
- Each length level has finitely many strings (2^n strings)
- We can order by length: 0, 1, 2, 3, ...

Therefore, {0,1}* is countable. ✓

EXAMPLE 4: Is ℕ×ℕ×ℕ (triples) countable? (7 minutes)

ANSWER: YES, ℕ×ℕ×ℕ is countable. ✅

JUSTIFICATION:

Enumerate by sum (i+j+k):

Sum 0: (0,0,0)
Sum 1: (0,0,1), (0,1,0), (1,0,0)
Sum 2: (0,0,2), (0,1,1), (0,2,0), (1,0,1), (1,1,0), (2,0,0)
...

Every triple (i,j,k) appears at sum level i+j+k.
Each level is finite.

Therefore, ℕ×ℕ×ℕ is countable. ✓

ALTERNATIVE: Since ℕ×ℕ is countable, and (ℕ×ℕ)×ℕ is countable,
ℕ×ℕ×ℕ is countable. (Countable × Countable = Countable)

TIME: ~10 minutes | TARGET: 10/10 points (EASIEST - FULL POINTS!)

==
PROBLEM 4: DECIDABILITY/RECOGNIZABILITY (20 points) - DO SECOND!
==

EXAMPLE 1: ALL_DFA (12 minutes)

QUESTION: Is ALL_DFA = {⟨A⟩ | A is DFA and L(A) = Σ*} decidable?

Part (a): Is ALL_DFA decidable?

ANSWER: YES, ALL_DFA is decidable. ✅

JUSTIFICATION:

To decide if a DFA accepts all strings:

1. Take input ⟨A⟩ (DFA encoding)
2. Construct complement DFA A':
   - Flip accept and non-accept states
   - L(A') = Σ* - L(A)
3. Check if L(A') = ∅ using E_DFA (emptiness test):
   - Run graph search from start state

  - Check if any accept state reachable
4. Decide:
  - If L(A') = ∅ → L(A) = Σ* → ACCEPT
  - If L(A') ≠ ∅ → L(A) ≠ Σ* → REJECT

Since E_DFA is decidable (graph reachability always terminates) and DFAs are closed under complement, this procedure always halts.

Therefore, ALL_DFA is decidable. ✓

Part (b): Is ALL_DFA Turing-recognizable?

ANSWER: YES, ALL_DFA is Turing-recognizable. ✅

JUSTIFICATION:

Since ALL_DFA is decidable (from part a), it is also recognizable.

KEY THEOREM: Every decidable language is Turing-recognizable.

Reasoning: If a TM always halts (decidable), it certainly halts and accepts when the answer is YES (recognizable).

Therefore, ALL_DFA is Turing-recognizable. ✓

EXAMPLE 2: EQ_CFG (THE TRICKY ONE!) (12 minutes)

QUESTION: Is EQ_CFG = {⟨G,H⟩ | L(G) = L(H)} decidable?

Part (a): Is EQ_CFG decidable?

ANSWER: NO, EQ_CFG is NOT decidable. ❌

JUSTIFICATION:

CFG equality cannot be decided because:

1. CFGs generate potentially infinite languages
2. Cannot exhaustively test all strings (infinite!)
3. CFGs NOT closed under intersection or complement (unlike DFAs)
4. Cannot minimize CFGs to canonical form (unlike DFAs)

Since there is no algorithmic way to compare two potentially infinite languages generated by CFGs, no TM can always decide this problem.

Therefore, EQ_CFG is NOT decidable. ✗

Part (b): Is EQ_CFG Turing-recognizable?

ANSWER: NO, EQ_CFG is NOT Turing-recognizable. ❌

JUSTIFICATION:

This is special! The COMPLEMENT is recognizable:

EQ̄_CFG (inequality) IS recognizable because:
- Enumerate all strings: ε, 0, 1, 00, 01, 10, 11, ...
- Test each in both languages using A_CFG (decidable)
- If find string in one but not other → ACCEPT (different!)
- This WILL find a difference if one exists

KEY THEOREM: If both L and L̄ are recognizable, then L is decidable.

Applying theorem:
- EQ̄_CFG IS recognizable ✓
- EQ_CFG is NOT decidable (from part a) ✗
- By theorem: if EQ_CFG were recognizable → would be decidable
- But it's NOT decidable!
- Therefore, EQ_CFG must NOT be recognizable

Therefore, EQ_CFG is NOT Turing-recognizable. ✗

EXAMPLE 3: A_TM (12 minutes)

QUESTION: Is A_TM = {⟨M,w⟩ | M accepts w} decidable?

Part (a): Is A_TM decidable?

ANSWER: NO, A_TM is NOT decidable. ❌

JUSTIFICATION:

We cannot decide if an arbitrary TM accepts a given input because:

1. TM might loop forever (we can't tell if it's looping or just slow)
2. No way to predict TM behavior in advance
3. This would solve the halting problem (known to be undecidable)

Classic proof: If we could decide A_TM, we could decide HALT_TM (known impossible).

Therefore, A_TM is NOT decidable. ✗

Part (b): Is A_TM Turing-recognizable?

ANSWER: YES, A_TM is Turing-recognizable. ✅

JUSTIFICATION:

To recognize A_TM:

1. Take input ⟨M,w⟩
2. Run M on input w (simulate it)
3. If M accepts w → ACCEPT
4. If M rejects w → REJECT
5. If M loops forever → we loop forever too (that's OK for recognizable!)

This recognizes A_TM: accepts exactly when M accepts w.

Note: We can't DECIDE it (might loop), but we CAN RECOGNIZE it
(accepts when answer is YES).

Therefore, A_TM is Turing-recognizable. ✓

QUICK REFERENCE FOR OTHER PROBLEMS:

E_DFA (DFA emptiness):
- Decidable? YES ✅ (graph reachability)
- Recognizable? YES ✅

EQ_DFA (DFA equality):
- Decidable? YES ✅ (minimize both, compare)
- Recognizable? YES ✅

A_CFG (CFG membership):
- Decidable? YES ✅ (parse finite derivations)
- Recognizable? YES ✅

E_CFG (CFG emptiness):
- Decidable? YES ✅ (check if generates anything)
- Recognizable? YES ✅

HALT_TM (TM halting):
- Decidable? NO ❌ (classic undecidable)
- Recognizable? NO ❌ (complement not recognizable)

E_TM (TM emptiness):
- Decidable? NO ❌ (can't tell if accepts nothing)
- Recognizable? NO ❌ (complement not recognizable)

TIME: ~25 minutes total | TARGET: 16/20 points (both parts partial credit OK)

==
==========
FINAL STRATEGY - READ THIS RIGHT BEFORE EXAM!
==
==========

DO PROBLEMS IN THIS ORDER:
1. Problem 3 (Countability) - 10 min - EASIEST! 10/10 ✓
2. Problem 4 (Decidability) - 25 min - Use lookup table! 16/20 ✓
3. Problem 2 (Binary Mod) - 20 min - English algorithm = 5 free! 13/15 ✓
4. Problem 1 (TM Design) - 15 min - Write examples first! 14/20 ✓
5. Review - 10 min