CS370: Computation and Complexity

Comprehensive Exam Study Notes

Based on January 2025, Autumn 2025, and Sample 2026 Exam Papers

Maynooth University

Contents

1 Exam Structure Overview

The CS370 exam consists of **7 questions**, all mandatory, each worth equal marks (approximately 10 marks each). The structure is highly consistent:

1. **Q1:** Countability Proof (enumerate a set using a TM)
2. **Q2:** Decidability Proof (construct a decider TM)
3. **Q3:** Undecidability Proof (mapping reduction from HALT)
4. **Q4:** Complement Not Turing-Recognizable
5. **Q5:** Prove language is in P *or* NP-hard
6. **Q6:** Prove language is in NP (polynomial-time verifier)
7. **Q7:** NP-Completeness (reduce from 3-SAT) + worked example

Key Point

Time allowed: 2 hours. One A4 sheet of handwritten notes permitted.

2 Question 1: Countability Proofs

2.1 Core Concept

A set S is **countable** if there exists a bijection $f : \mathbb{N} \to S$, i.e., we can list all elements of S without repetition.

To prove countability, construct a Turing Machine that **enumerates** all elements of the set in some order (typically lexicographic).

2.2 The Dovetailing Technique

Definition 2.1 (Dovetailing)**.** Systematically iterate through all possible strings over an alphabet in lexicographic order:

$$\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots$$

For each string, test if it belongs to the set. If yes, output it.

2.3 General Template

Exam Template

M_1 = "On any input (or "On input n"):

1. Set counter $c = 0$ (or $c = 1$ for file numbering)
2. For each string s in lexicographic order over the appropriate alphabet:
   (a) Write s to a temporary file f
   (b) Run the **decider program** on file f
   (c) If the decider accepts (file is valid):
   - *Variant A (enumerate all):* Output s to file c.ext, increment c
   - *Variant B (index n):* If $c == n$, return s. Else increment c

"

Since M_1 enumerates the set without skipping elements, this proves the set is countable.

2.4 Exam Variations

2.4.1 January 2025: LibreOffice Writer Documents

- Set: $\{\langle D \rangle : D$ is a valid LibreOffice Writer document$\}$
- Decider: `writerlo` command-line program
- Output files: `a.odt, aa.odt, aaa.odt, ...`
- Alphabet: Binary (documents are binary files)

2.4.2 Autumn 2025: Python Programs

- Set: $\{\langle P \rangle : P$ is a valid Python program$\}$
- Decider: `python` command (syntax check)
- Input: integer n, output: the nth Python program
- Alphabet: Keyboard characters (text files)

2.4.3 Sample 2026: JPEG Files

- Set: $\{\langle J \rangle : J \text{ is a valid JPEG file}\}$
- Decider: `jpegzero` command
- Output files: `1.pdf, 2.pdf, 3.pdf, ...`
- Alphabet: Binary (bytes)

Common Mistake

The key insight is that a **decider exists** for file format validity. This is always decidable because:

- File formats have finite, well-defined specifications
- A program can check headers, structure, and syntax in finite time

3 Question 2: Decidability Proofs

3.1 Core Concept

A language L is **decidable** if there exists a TM D that:

- Accepts every $w \in L$
- Rejects every $w \notin L$
- Always halts

3.2 Key Decidable Problems

Theorem 3.1. Any property of **finite automata** (DFA/NFA) is decidable, including:

- Does FA M accept word w?
- Does FA M accept any word?
- Does FA M accept a word of length $> k$?
- Does FA M accept a word starting with symbol σ?
- Is $L(M_1) = L(M_2)$?

Proof Idea. FAs have finite state spaces. We can simulate them or analyze their structure in finite time. □

3.3 Exam Template

Exam Template

Proof: We will prove L is decidable by constructing a TM D to decide L.
D = "On input $\langle \ldots \rangle$:

1. Perform finite computation/simulation
2. If [condition satisfied]: **accept**
3. Else: **reject**"

Since D always halts and correctly decides membership in L, this proves L is decidable.

3.4 Exam Variations

3.4.1 January 2025: FA Accepting Words of Length > 5

$$L_2 = \{\langle M \rangle : M \text{ is a FA that accepts at least one word } w \text{ where } |w| > 5\}$$

Solution Strategy:

1. Construct FA M' that accepts $\Sigma^{>5} = \{w : |w| > 5\}$
2. Compute intersection $L(M) \cap L(M')$
3. Test if this intersection is non-empty (decidable for FAs)
4. Alternative: Use pumping lemma reasoning—if M has n states and accepts any word of length $> n$, it accepts infinitely many words of arbitrary length

3.4.2 Autumn 2025: FA Accepting Words Beginning with 0

$$L_2 = \{\langle M \rangle : M \text{ is a FA and } \exists w \in L(M) \text{ where first symbol of } w \text{ is } 0\}$$

Solution:

1. Construct FA M' accepting $0\Sigma^*$ (words starting with 0)
2. Compute $L(M) \cap L(M')$
3. Test if intersection is non-empty

3.4.3 Sample 2026: TM in State s After t Transitions

$$L_2 = \{\langle M, s, t \rangle : M \text{ is a TM, and when run on } \epsilon, M \text{ is in state } s \text{ after exactly } t \text{ transitions}\}$$

Solution:

1. Simulate M on empty input for exactly t steps
2. Check if current state equals s
3. Accept if yes, reject otherwise

Key Point

This is decidable because we only simulate for a **fixed, finite** number of steps t. We don't wait for M to halt—we just observe its state after exactly t transitions.

4 Question 3: Undecidability via Mapping Reduction

4.1 Core Concept

Definition 4.1 (Mapping Reduction)**.** Language A is **mapping reducible** to language B, written $A \leq_m B$, if there exists a computable function $f : \Sigma^* \to \Sigma^*$ such that:

$$w \in A \iff f(w) \in B$$

Theorem 4.1. If $A \leq_m B$ and A is undecidable, then B is undecidable.

4.2 The Halting Problem

Definition 4.2 (HALT)**.**

$$\text{HALT} = \{\langle M, w\rangle : M \text{ is a TM and } M \text{ halts on input } w\}$$

HALT is undecidable (given).

4.3 Reduction Template

Exam Template

Proof: We will use a mapping reduction to prove HALT $\leq_m L_3$.
Assume that L_3 is decidable.
The transition function f that maps instances of HALT to instances of L_3 is given by TM F:
F = "On input $\langle M, w\rangle$:

1. Construct the following N given by the following pseudocode:

   N = "On input u:

   (a) Optional: Check format of u, reject if wrong format

   (b) Run M on w

   (c) Accept"

2. Return $\langle N\rangle$"

Now, $\langle N\rangle \in L_3$ iff $\langle M, w\rangle \in$ HALT.
So, using f and the assumption that L_3 is decidable, we can decide HALT. Contradiction.
Therefore, L_3 is undecidable.

4.4 Understanding the Construction

The key insight is:

- N **ignores** its input u
- N first runs M on w (embedded in N's code)
- If M halts on w: N accepts (so N accepts everything, including words matching the property)
- If M loops on w: N never reaches the accept step (so N accepts nothing)

4.5 Exam Variations

4.5.1 January 2025: TM Accepts Word of Length > 5

$$L_3 = \{\langle M \rangle : M \text{ accepts at least one word } w \text{ where } |w| > 5\}$$

N = "On input u:

1. If $|u| \leq 5$: reject
2. Run M on w
3. Accept"

4.5.2 Autumn 2025: TM Accepts Word Starting with 0

$L_3 = \{\langle M \rangle : M \text{ is a TM with } \Sigma = \{0, 1\} \text{ that accepts at least one word starting with } 0\}$

N = "On input u:

1. If u does not begin with 0: reject
2. Run M on w
3. Accept"

4.5.3 Sample 2026: TM Accepts At Least 5 Words

$$L_3 = \{\langle M \rangle : |L(M)| \geq 5\}$$

N = "On input u:

1. Run M on w
2. Accept"

(If M halts on w, N accepts everything, so $|L(N)| = \infty \geq 5$. If M loops, $|L(N)| = 0 < 5$.)

Common Mistake

Common mistake: Forgetting that M and w are **hardcoded** into N. The reduction function F outputs a complete description of N that contains the encoding of M and w.

5 Question 4: Complement Not Turing-Recognizable

5.1 Core Concept

Theorem 5.1. If L is Turing-recognizable but not decidable, then $\overline{L}$ (the complement of L) is **not** Turing-recognizable.

Proof. Suppose both L and $\overline{L}$ are T-recognizable. Then we could decide L by running recognizers for L and $\overline{L}$ in parallel—one must accept. This contradicts that L is undecidable. □

5.2 Strategy

To show $\overline{L_3}$ is not T-recognizable:

1. Show L_3 **is** T-recognizable (construct a recognizer)
2. Note that L_3 is undecidable (from Q3)
3. Conclude $\overline{L_3}$ is not T-recognizable

5.3 Template for Recognizing L_3

Exam Template

Proof: We prove the complement of L_3 is not T-r by constructing a TM M_4 that recognizes L_3.
M_4 = "On input $\langle M \rangle$:

1. For each word $w \in \Sigma^*$ (in some enumeration):
   (a) Optional: If w matches the required property (e.g., starts with 0, length > 5):
   (b) Run M on w in parallel (breadth-first simulation)
   (c) If any running instance of M accepts: **accept**

"
TM M_4 recognizes L_3. Since we proved in Q3 that L_3 is undecidable, the complement of L_3 is not T-recognizable.

5.4 Parallel/Breadth-First Simulation

Definition 5.1 (Dovetailing for TM Simulation)**.** To run infinitely many TM instances in parallel:

1. Round 1: Run M on w_1 for 1 step
2. Round 2: Run M on w_1 for 2 steps, M on w_2 for 1 step
3. Round 3: Run M on w_1 for 3 steps, w_2 for 2 steps, w_3 for 1 step
4. Continue...

If any instance accepts, we detect it in finite time.

Key Point

This is a **recognizer**, not a decider. If no word is accepted, M_4 runs forever. That's okay—recognizers are allowed to loop on non-members.

6 Question 5: P vs NP-Hard

6.1 Core Definitions

Definition 6.1 (Class P)**.** $L \in$ P if there exists a TM T that decides L in time $O(n^k)$ for some constant k, where n is the input size.

Definition 6.2 (NP-Hard)**.** L is NP-hard if every problem in NP can be reduced to L in polynomial time.

6.2 Proving $L \in$ P

Exam Template

Proof: We prove $L_5 \in$ P by constructing a TM T to recognize L_5 in polynomial time.
T = "On input $\langle \cdot \rangle$:

1. Algorithm steps with complexity annotations
2. Accept/Reject based on result"

Complexity Analysis:

- Step 1: $O(\cdot)$
- Step 2: $O(\cdot)$
- ...

Total: O(polynomial)
Since T recognizes L_5 in polynomial time, $L_5 \in$ P.

6.3 Exam Variations

6.3.1 January 2025: Common Element in All Sets

$$L_5 = \{\langle L \rangle : L \text{ is a collection of } M \text{ sets, all sharing at least one common integer}\}$$

Algorithm:

1. For each integer a in the first set: $[N]$
2. Initialize counter $c = 0$ [1]
3. For each set t in L: $[M]$
4. For each integer b in t: $[N]$
5. If $a == b$: increment c, break [1]
6. If $c == M$: accept [1]
7. Reject

Complexity: $O(N \cdot M \cdot N) = O(MN^2)$ — polynomial!

6.3.2 Autumn 2025: Graph Edge Symmetry

$$L_5 = \{\langle G \rangle : G = (V, E) \text{ where } \forall (a, b) \in E, (b, a) \in E\}$$

Algorithm:

1. For each edge (a, b) in E: $[|E|]$
2. If $(b, a) \notin E$: reject $[|E|]$ (lookup)
3. Accept

Complexity: $O(|E|^2)$ or $O(|E|)$ with hash set — polynomial!

6.3.3 Sample 2026: Sets Covering 1–100

$$L_5 = \{\langle L \rangle : L \text{ is collection of sets that together contain all integers } 1, \ldots, 100\}$$

Algorithm:

1. Create boolean array found[1..100], initialize to false
2. For each set s in L: $[M]$
3. For each integer x in s: $[N]$
4. If $1 \leq x \leq 100$: found$[x]$ = true $[1]$
5. If all entries in found are true: accept
6. Reject

Complexity: $O(MN + 100) = O(MN)$ — polynomial!

Key Point

To show something is in P, give an explicit algorithm and count the operations. Use big-O notation with input size parameters like M, N, $|V|$, $|E|$.

7 Question 6: Proving a Language is in NP

7.1 Core Concept

Definition 7.1 (Class NP)**.** $L \in$ NP if there exists a polynomial-time **verifier** V such that:

$$w \in L \iff \exists \text{ certificate } c \text{ such that } V(w, c) \text{ accepts}$$

The certificate c has polynomial size in $|w|$.

7.2 Template

Exam Template

Proof: We prove $L_6 \in$ NP by constructing a polynomial-time verifier M_6.
Certificate: $c =$ [describe what the certificate is]
M_6 = "On input $\langle$instance$, c\rangle$:

1. Verify c has correct format/size
2. Check that c satisfies the required property
3. Accept if all checks pass, reject otherwise"

Complexity Analysis: [Show each step is polynomial]
Since M_6 verifies L_6 in polynomial time, $L_6 \in$ NP.

7.3 The Hitting Set Problem

All Q6 variations are essentially the **Hitting Set** problem:

Definition 7.2 (Hitting Set)**.** Given a collection of sets $\{S_1, \ldots, S_n\}$ and integer k, does there exist a set H of size $\leq k$ that "hits" every set (i.e., $H \cap S_i \neq \emptyset$ for all i)?

7.4 Exam Variations

7.4.1 January 2025: Students Covering Exams

- **Sets:** $E_1, \ldots, E_x$ (students who attended each exam)
- **Question:** Can we pick k students to cover all exams?
- **Certificate:** The set of k students

7.4.2 Autumn 2025: Players Covering Matches

- **Sets:** $M_1, \ldots, M_n$ (players in each match)
- **Question:** Can coach pick k players covering all matches?
- **Certificate:** The set of k players

7.4.3 Sample 2026: Teacher Guessing Numbers

- **Sets:** $A_1, \ldots, A_N$ (each student's list of P numbers)
- **Question:** Can teacher guess k numbers hitting every list?
- **Certificate:** The set of k numbers

7.5 Verification Algorithm

Exam Template

M_6 = "On input $\langle \{S_1, \ldots, S_n\}, k, c \rangle$ where c is the proposed hitting set:

1. Check $|c| \leq k$ $[O(k)]$
2. For each set S_i: $[n]$
   (a) For each element x in c: $[k]$
   (b) If $x \in S_i$: mark S_i as hit, break $[|S_i|]$
3. If all sets are hit: accept
4. Reject"

Complexity: $O(n \cdot k \cdot m)$ where m is max set size — polynomial!

8 Question 7: NP-Completeness via 3-SAT Reduction

8.1 Core Concept

Definition 8.1 (NP-Complete)**.** L is NP-complete if:

1. $L \in$ NP
2. L is NP-hard (every NP problem reduces to it)

Theorem 8.1. To prove L is NP-complete:

1. Show $L \in$ NP (done in Q6)
2. Show a known NP-complete problem reduces to L in polynomial time

8.2 3-SAT Definition

Definition 8.2 (3-SAT)**.** Given a Boolean formula in CNF where each clause has exactly 3 literals:

$$C = (l_1 \vee l_2 \vee l_3) \wedge (l_4 \vee l_5 \vee l_6) \wedge \cdots$$

Is there a truth assignment satisfying all clauses?

8.3 3-SAT to Hitting Set Reduction

Theorem 8.2. 3-SAT $\leq_p$ Hitting Set

Construction. Given 3-SAT formula C with d clauses and g variables:

1. For each clause $(l_1 \vee l_2 \vee l_3)$, create a set $\{l_1, l_2, l_3\}$
2. Set $k = g$ (number of variables)
3. Output $\langle \{S_1, \ldots, S_d\}, k \rangle$

□

Key Point

The hitting set corresponds to choosing one literal per clause. Setting $k = g$ ensures we pick at most one literal per variable (either x or $\bar{x}$, not both).

8.4 Reduction Template

Exam Template

Proof: We prove NP language L_6 is NP-complete by constructing a polynomial-time reduction from 3-SAT.
F = "On input $\langle C \rangle$ where C has d clauses and g variables:

1. For each clause $C_i = (l_1 \vee l_2 \vee l_3)$ in C:
   (a) Create set $S_i = \{l_1, l_2, l_3\}$
2. Set $k = g$
3. Return $\langle \{S_1, \ldots, S_d\}, k \rangle$"

Correctness: $\langle \{S_1, \ldots, S_d\}, k \rangle \in L_6$ iff $\langle C \rangle \in$ 3-SAT.
Complexity: $O(d \cdot 3) = O(d)$ — polynomial!

8.5 Worked Example (Sample 2026)

Input: $C = (a \vee b \vee c) \wedge (b \vee d \vee e) \wedge (d \vee e \vee f) \wedge (b \vee e \vee f)$
Variables: a, b, c, d, e, f (so $g = 6$)
Clauses: 4 (so $d = 4$)
Reduction:

$$S_1 = \{a, b, c\}$$
$$S_2 = \{b, d, e\}$$
$$S_3 = \{d, e, f\}$$
$$S_4 = \{b, e, f\}$$

Output: $\langle \{\{a, b, c\}, \{b, d, e\}, \{d, e, f\}, \{b, e, f\}\}, 6 \rangle$

Key Point

The literals in the sets are the actual literals (including negations like $\bar{a}$), not just variables. If a clause contains $\bar{x}$, use $\bar{x}$ in the set.

9 Quick Reference: Key Theorems

1. **Countable sets:** Can be enumerated by a TM
2. **Decidable:** TM always halts with correct answer
3. **FA properties:** All decidable (finite state space)
4. **TM properties:** Generally undecidable (Rice's theorem)
5. **HALT:** Undecidable (given)
6. **If $A \leq_m B$ and A undecidable:** B undecidable
7. **T-recognizable but undecidable:** Complement is not T-r
8. **P:** Decidable in polynomial time
9. **NP:** Verifiable in polynomial time with certificate
10. **NP-complete:** In NP and NP-hard
11. **3-SAT:** NP-complete (given)
12. **Hitting Set:** NP-complete (prove via 3-SAT)

10 Common Mistakes to Avoid

Common Mistake

1. **Q1:** Forgetting that the decider (writerlo, python, etc.) is given and always halts
2. **Q2:** Confusing decidability of FA properties vs TM properties
3. **Q3:** Forgetting that M and w are hardcoded into N
4. **Q3:** Writing "Run M on u" instead of "Run M on w"
5. **Q4:** Trying to build a decider instead of a recognizer
6. **Q5:** Not showing complexity analysis
7. **Q6:** Forgetting to specify the certificate
8. **Q7:** Confusing variables with literals in the reduction
9. **Q7:** Setting k incorrectly (should be number of variables)

11 Last-Minute Checklist

Before the exam, make sure you can:

- ☐ Write a dovetailing enumeration algorithm
- ☐ Construct decidability proofs for FA properties
- ☐ Fill in the mapping reduction template from HALT
- ☐ Construct a recognizer using parallel simulation
- ☐ Analyze algorithm complexity with big-O notation
- ☐ Define appropriate certificates for NP problems
- ☐ Reduce 3-SAT to Hitting Set
- ☐ Work through a 3-SAT reduction example by hand

Good luck with your exam!