Final Exam: Study Guide

CSE 30151 Spring 2016

2016/05/06

The exam will be on Friday, May 6, 10:30am to 12:30pm, in 126 DeBartolo Hall (same as lectures). It will be open book and open paper notes. No computers, smartphones, or any other Turing-equivalent machines are allowed. Regrettably, I can't think of any way to allow the use of notes taken on an electronic tablet that is fair to all students.

Format

The exam is worth 120 points, or 20% of your grade. It covers the entire course, but not any of these special topics: neural networks and finite automata, human language and context-free grammars, human intelligence and Turing machines, cryptography.

The main part of the exam will present you with five languages, one from each of the following classes:

I. Regular

II. Context-free but not regular

III. In P but not context-free

IV. NP-complete

V. Turing-recognizable but not decidable.

For each language, you'll identify which class it belongs to (2 points each) and justify your answer. Your justifications should have the following forms:

- Regular
  - A DFA, NFA, or regular expression (10 points; like HW2 Q1, HW4 Q2a, HW4 Q3a, Exercise 1.6j, 1.18e)
- Context-free but not regular
  - A PDA or CFG (10 points; like HW5 Q1–2, Exercise 2.4ad, 2.6ac, 2.7ac)

  - A proof of non-regularity (10 points; like HW 4 Q2b, Q3b, Problem 1.29ac, 1.46b)
- In P but not context-free
  - An implementation-level description of a TM and a brief time complexity analysis (10 points; like HW7 Q1–2, Exercise 3.8a)
  - A proof of non-context-freeness (10 points; like HW6 Q1, Problem 2.30bc)
- NP-complete (like Problem 7.22, 7.31, HW9 Q4, but easier)
  - A high-level description of a NTM or verifier (5 points)
  - A polynomial-time reduction from another NP-complete problem and a proof that it works (15 points)
- Turing-recognizable but not decidable (like HW8 Q3, Problems 5.10, 5.11)
  - A high-level description of a TM (5 points)
  - A reduction from another undecidable language and a proof that it works (15 points)

The remaining 20 points will be for a question or questions related to Turing machines and the Church-Turing thesis.

Sample questions

Here's an example of five languages that you should be able to classify as (I) regular, (II) context-free but not regular, (III) in P but not context-free, (IV) NP-complete, or (V) Turing-recognizable but not decidable.

1. Classify:

$$A = \{x\texttt{=}y\texttt{+}z \mid x, y, z \text{ are binary natural numbers and } x = y + z \text{ is true}\}$$

2. A *0-1 integer program* is a system of inequalities of the form:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &\le b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &\le b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &\le b_m \end{aligned}$$

   where the a_{ij} and b_i are integers, and the x_j are variables. A solution to the 0-1 IP is a setting of each of the x_j to **either 0 or 1** such that all of the inequalities hold.

   Classify: The set B of all (encodings of) 0-1 integer programs that have solutions.

3. Classify: The set C of all (encodings of) Turing machines that, on empty input, halt with nothing but the string `42` on the tape.

4. Classify:
$$D = \{x \mid x \text{ is a binary natural number divisible by } 3\}$$

5. Let $\Sigma = \{\mathtt{1}, +, -, *, /\}$. We say that a string $w \in \Sigma^*$ is an *RPN expression* if typing the symbols of w into an RPN calculator results in no stack underflows and a stack with a single number. For example, `11+` is an RPN expression, but `1+` is not (stack underflow) and `111+` is not (results in more than one number on stack).

   Classify: the language E of all RPN expressions.

Sample (partial) solutions

1. This language is in P but not context-free. (In HW4 Q2b you showed that it was not regular, but it's also not context-free.)

2. This language is NP-complete.

   - An NTM can solve the system of inequalities by nondeterministically trying all possible settings of the x_j. For each setting, checking whether all the inequalities hold can be done in $O(mn)$ time. So this language is in NP.
   - We want to show that if we could solve a 0-1 IP in polynomial time, then we could solve 3-SAT in polynomial time.

     So, given a formula ϕ in 3-CNF, we want to convert it to a 0-1 IP. The formula ϕ is of the form

     $$\phi = (\phi_{11} \lor \phi_{12} \lor \phi_{13}) \land \cdots \land (\phi_{m1} \lor \phi_{m2} \lor \phi_{m3}),$$

     where each ϕ_{ik} is either x_j or $\overline{x_j}$ for some j. The m clauses of ϕ become m inequalities: $\lor$ becomes $+$, x_j becomes y_j, and $\overline{x_j}$ becomes $(1 - y_j)$, resulting in an arithmetic expression which we require to be ≥ 1. For example, the formula

     $$(x_1 \lor x_1 \lor x_2) \land (\overline{x_1} \lor \overline{x_2} \lor \overline{x_2}) \land (\overline{x_1} \lor x_2 \lor x_2)$$

     becomes the 0-1 IP

     $$\begin{aligned} y_1 + y_1 + y_2 &\geq 1 \\ (1 - y_1) + (1 - y_2) + (1 - y_2) &\geq 1 \\ (1 - y_1) + y_2 + y_2 &\geq 1. \end{aligned}$$

     A little bit of algebra can put this system of inequalities into the form in the definition of 0-1 IP. All this can be done in $O(m)$ time.

Now, if we know a solution to this 0-1 IP, then we can come up with a satisfying assignment for ϕ very easily: if $y_j = 0$, then x_j is false, and if $y_j = 1$, then x_j is true. In each of the inequalities, at least one of the addends must be nonzero, which means that in each clause of ϕ, at least one of the literals must be true, so ϕ is satisfied.

But if the 0-1 IP has no solution, then ϕ is unsatisfiable. For if there were a setting of the x_j that satisfied ϕ, then we could come up with a solution for the 0-1 IP: if x_j is false, then $y_j = 0$, and if x_j is true, then $y_j = 1$. In each of the clauses, at least one of the literals must be true, which means that in each inequality of the 0-1 IP, at least one of the addends must be nonzero, so the 0-1 IP is solved.

3. This language is Turing-recognizable but not decidable – it's only trivially different from A_{TM}.

   - It is recognizable by the TM that, on input $\langle M \rangle$, does:
     (a) Simulate M with the empty string as input.
     (b) If it halts and the tape reads `42`, accept. Otherwise, reject.
   - Suppose C were decidable by a TM R. For any TM M and string w, we can construct a TM M' that does:
     (a) Run M on w.
     (b) If M accepts w, clear the tape, write `42`, and accept.
     (c) Otherwise, clear the tape and accept.

     Then we could construct a TM S that, on input $\langle M, w \rangle$, does:
     (a) Construct a TM M' as described above.
     (b) Run R on M'.
     (c) If R accepts M', accept.
     (d) Otherwise, reject.

     If M accepts w, then M' prints `42`, so R accepts M', so S accepts $\langle M, w \rangle$. On the other hand, if M rejects w or loops, then M' prints nothing or loops, so R rejects M', so S rejects $\langle M, w \rangle$. Thus, S decides A_{TM}. But this contradicts the fact that A_{TM} is undecidable. Therefore, C is undecidable.

4. This is a regular language (HW2 Q1).

5. This language is context-free but not regular.