Final Exam Preparation

Final Exam

The final exam is scheduled by the registrar for **Saturday, 10 December, 9am-noon** in our normal classroom. The final will cover everything in the course, with an emphasis on the most important concepts that have appeared in at least two places.

Most of the questions on the final will be small variations on problems you have already seen on previous exams or problem sets. Doing well on these questions will make a strong case for earning at least a B in the class. A few of the problems will be designed to see how well you can use concepts you have learned in the class to solve problems unlike ones you have already seen. Doing well on many of these questions will make a strong case for earning an A in the class.

The format will be fairly similar to Exam 1 and Exam 2, but because of the extended time for the final, and the desire to give as much opportunity as possible for students to demonstrate what you can do, will be a bit longer than those exams.

As with Exam 1 and Exam 2, you will be permitted to use a **single paper page of notes that you prepare and bring to the exam**, but no other resources. It is fine to collaborate with others to prepare your notes. The page should be no larger than a US Letter size page (8.5×11 inches), and you may write (or print) on both sides of the page.

Unlike the previous exams, you must turn in your notes page with your exam. If you would like to keep your own copy of it, you should make a copy to save before the exam.

Expected Problems

Although the exam covers the whole class, you should not be surprised if it includes problems for you to demonstrate:

– Your understanding of **logical formulas**, **inference rules**, and an ability to reason about and manipulate formulas using **quantifiers**.

– Your fluency with standard proof techniques including **proof-by-contradiction**.

– Your understanding of **well-ordering** and how to construct proofs using the **well ordering principle**.

– Your understanding of **sets**, how the set operators are defined, and what they mean.

– That you can do a **regular induction** proof similar to ones you have seen on previous exams *very well.* That you can do an induction proof that requires some creativity to **define a good induction predicate** and then to complete the proof.

– Your understanding of **state machines**, and ability to use the **invariant principle** to prove a property of the reachable states for a given state machine.

– Your understanding of **recursive data types**, ability to define functions on recursive data types, and to use **structural induction** to prove a property about all objects of a recursive data type.

– Your understanding of infinite cardinalities including the ability to determine if a set is **countable** or **uncountable**, and to support your answer with a convincing proof.

– Your understanding of **Turing Machines** and **computability**. (See practice problems.)

Practice Problems

Since there was no problem set covering Classes 23–25, here are some practice problems to help you prepare for the final. You do not need to turn in your solutions to these problems, but it is highly recommended that you approach them similarly to a problem set to be well prepared for the final. I will post solutions to the problems on ths course site by December 4, but it is strongly encouraged that you solve the problems on your own before consulting the solutions. You should not be surprised to see problems similar to these on the final exam.

1. Prove that all well-ordered sets are countable.
2. The way we defined an execution of a Turing Machine in Class 23 suggested that there could be more than one execution of a Turing Machine, $TM = (S, T \subseteq S \times \Gamma \rightarrow S \times \Gamma \times \textit{dir}, q_0 \in S, q_{Accept} \subseteq S)$, on a given input I. What property of T would ensure that there is only a single execution possible?

For the following questions, a *deterministic* Turing Machine is a Turing Machine that for every possible input has a single execution. That is, there is no input for which the Turing Machine has more than one possible execution.

3. Describe a deterministic Turing Machine that never halts but never repeats a configuration.
4. Prove that a deterministic Turing Machine that repeats a configuration never halts.

Consider the Turing Machine, M_X, defined below. (The **-** symbol denotes a blank square, which may not appear in the input. Every square to the right of the last input symbol is initially blank.) Hint: for questions 5 and 6 you should be able to use similar techniques to how we reasoned about state machines, but need to also take into account the tape (so instead of using the invariant principle for states as before, now you need to consider it for full machine configurations).

$$\begin{aligned} M_X = (S = \{A, B, C, D\}, & \\ T = \{ & (A, \mathbf{0}) \rightarrow (B, \mathbf{X}, \mathbf{R}), (A, \textbf{-}) \rightarrow (D, \textbf{-}, \textbf{Halt}) \\ & (B, \mathbf{0}) \rightarrow (B, \mathbf{0}, \mathbf{R}), (B, \textbf{-}) \rightarrow (C, \textbf{-}, \mathbf{L}), \\ & (C, \mathbf{0}) \rightarrow (C, \mathbf{0}, \mathbf{L}), (C, \mathbf{X}) \rightarrow (A, \mathbf{X}, \mathbf{R})\}, \\ q_0 = A, & \, q_{Accept} = \{D\}) \end{aligned}$$

5. Prove that M_X running on any initial tape with finite input (that is, the number of non-blank squares is $k \in \mathbb{N}$) always terminates.
6. Prove that $\mathcal{L}(M_X) = \mathbf{0}^*$ (that is, the language recognized by M_X is the set of all strings of zero or more **0** symbols).