Preparation for the Final Exam

Reading: The exam will be based on Sipser, Pinker, and any material covered in class that is not in either of these places:

- **Chapter 0**: Everything except for the sections on graphs (pp. 10–13), boolean logic (pp. 14–15), and induction (pp. 23-25).
- **Chapter 1**: All, except for pp. 70–76. That’s the pages on using GNFAs to convert NFAs to regular expressions; all you need to know is that it can be done.
- **Chapter 2.1**: All of it.
- **Chapters 2 and 4** from Pinker. You should be familiar enough with them to answer content questions. You should also have an understanding of the relevance and implications of automata theory for linguistics.
- Anything we did in class but is not in the book, in particular, how to convert REs into CFGs.

Cheat sheet: You may bring with you any handwritten notes, diagrams etc. you can fit on one full notebook page (written on one side only). No photocopies are allowed. If you prepare your sheet carefully, the preparation process will be an effective study aid.

Content: Pretty much what we have been doing in class. The exam will cover material from the entire semester, with more emphasis on the second half. (There’s no way to understand the material in the second half without command of the early stuff, anyway). In general, the amount of time we spent on something in class is a good predictor of its importance in the exam.

There will be a couple of general linguistics questions, some set theory problems, some questions related to formal definitions and concepts, some constructions and proofs. You will need to be very clear on the meaning of the terminology and the notation (your “cheat sheet” will help you cut down on memorizing, but it is no substitute for understanding). You will especially need to know how to design simple automata and grammars, and how to combine them using the various constructions we learned.

Preparation: Your best bet is to study your notes and the reading carefully, especially anything that you got wrong in the homework, or that didn’t make much sense the first time through. Test your understanding by re-working any past homeworks that you got wrong.

Key Concepts and Topics: A list is given in the following pages. You can use it as a study aid.
Key Concepts and Topics

1 Set Theory

Sets, finite and infinite, and the notation for specifying them. (E.g., what is $\{x \mid 5 \leq x \leq 10\}?$). Set operations: union, intersection, complement, power set, Cartesian product. Element vs. subset, sets of sets, etc. Ordered pairs and tuples. The standard sets $\mathbb{N}$, $\mathbb{Z}$.

Functions: Understand how to write and use function tables for one- and two-argument functions; domain and range; using functions.

Strings: $\varepsilon$; language, alphabet, length of a string.

2 Finite State Automata

Languages: Understand the meaning of languages given in set notation or in English; the difference between $\phi$ (or $\{ \}$) vs. $\varepsilon$ vs. $\{ \varepsilon \}$ vs. $\{0\}$ vs. 0.

FSA Basics: start state, accept and reject states; formal description (as a 5-tuple); the transition function; requirements for a well-formed FSA (no missing or multiple arrows, labels, etc).

Understanding FSAs: Given an automaton, you should be able to tell: how it works, and exactly how it will work if it is fed some string of my choice; what kinds of strings it accepts and what it rejects. Designing FSAs to order (see Addendum).

FSA Constructions (ways of making FSAs from other FSAs): Negating an FSA; the Cartesian product construction (Theorem 1.12) for the union or intersection of languages.

NFA Basics: formal description; differences between NFAs and DFAs; $\varepsilon$-transitions. Designing NFAs.

NFA Constructions: Converting an NFA to a DFA; building NFAs from smaller NFAs, using the regular operations.

Regular languages; the regular operations: $\cup$, $*$, $\circ$; closure.

3 More on Regular Languages

Regular Expressions. Reading and constructing. Conversion of REs to NFAs.

Regular Languages. Equivalence of DFAs, NFAs, and REs.

Non-Regular Languages. The infamous pumping lemma: What it says, why it’s true, how to use it.
4 Context-Free Grammars

**Basics:** how they work, how to write them.

**Constructions:** How to combine CFGs according to the regular operations; how to convert an RE into a CFG. *(Not in the book: Refer to your notes)*.

**Applications:** Using for linguistic grammars. Properties of English that make it non-regular. X-bar theory. (Pinker).

5 Addendum: Designing automata

Some problems are easier to solve using NFAs, some using DFAs. (You may be required to use one or the other, in which case you will have no choice). There are several different types of problems, an for each type there are some tricks of the trade for solving it.

**General Advice:**

1. Remember: it is just as important for your automaton to *reject* the bad strings as it is for it to accept the good ones!

2. Pay particular attention to what happens once you get to an accept state: if the string continues, do you accept it, reject it, or keep computing? The answer depends on the problem you are solving!

**Types of Languages** to Recognize:

1. Finite list of strings, e.g., \{ aba, bbab, ε, aaabb \}.

2. All strings that begin or end with some given string (e.g., with 01001).

3. All strings that contain some substring anywhere inside them.

4. Counting by length; e.g., all strings of length exactly/at least/at most five.

5. Counting something (e.g., all strings that contain exactly/at least/at most three zeros).

6. Counting even/odd (or multiples of three, etc): strings of odd length, strings with an even number of zeros, etc.

7. Combinations of the above: strings that begin with 1 and have odd length, etc.

Some such problems are best solved by writing small automata and combining them using various techniques.

That's It... Good Luck!!

3