Checklist of all the things you’ve learned in CS 3100, Fall 2010
Also a small Mapping Reduction proof at the end
Handed out 12/9/10

Final Exam: The final exam will have closed-book multiple-choice short questions on all these topics below, for 50 minutes. Then after a 5-minute break, you’ll be given a 60-minute long open-book exam on all the bold-faced topics plus the mapping reduction proof at the end (or small variants of this mapping reduction).

[ ] Designing simple FAs and Reg Exps
[ ] Identify strings in a given Reg Exp
[ ] Basic notions about sets and strings (Powerset etc)
[ ] FA, Reg Exp conversions
[ ] DFA to Reg Exp conversion
[ ] DFA minimization (studied much later)
[ ] FA operations (intersection, reversal, etc), and whether closure is guaranteed
[ ] Why some languages are not regular; Pumping lemma

[ ] Alternate characterization of regularity: Ultimate periodicity and “lasso shapes” for minimal DFA (studied much later) over a singleton alphabet
[ ] Midterm examined the above, esp. exp growth of NFA/DFA conversion, etc.
[ ] Flex experiments
[ ] PDA design using JFLAP; NPDA, DPDA
[ ] What JFLAP helps you do: freeze configurations, watch non-determinism evolve, Pumping Lemma tutor, conversions from DFA to RE, etc.

[ ] Designing simple CFGs
[ ] CFG consistency, completeness, simplification
[ ] Pumping Lemma for CFLs
[ ] Why certain CFLs are not closed under complementation
[ ] Parsing using dynamic programming using the Chomsky normal form of a CFG (the table filling idea)
[ ] CFG to PDA and back
[ ] The Chomsky normal form; why it guarantees certain derivation lengths
[ ] General story of pumping: not an iff theorem
[ ] Yacc based design of calculator
[ ] Linearity of CFGs, and what it means
PDA and CFG operations (union, intersection, etc.) and whether closure is guaranteed

The LBA classification (briefly) and context sensitive languages

Designing simple Turing machines (DTM, NDTM, multi-tape TM).

Language classifications: RE, Recursive, etc. and what it means

Basic results: Universality of CFGs being undecidable; emptiness being decidable; status of grammar equivalence (decidable or not)

Printer TM and decider TM, conversions

Self referential statements, self-denying TMs $S_{TM}$ being undecidable

Favorite sets: $A_{TM}$ etc. and status of decidability

Diagonalization. Use in cardinality comparison

Notion of onto and into functions

Schröder-Bernstein Theorem and its uses to compare cardinalities

Cardinality based arguments to show there are non-RE languages

Proof of undecidability of the Halting problem. Two approaches: diagonalization proof, and proof based on $S_{TM}$.

Time-complexity classes NP, P, NPC. What a non-deterministic algorithm is.

Mapping reductions: what they are.

Mapping reductions for showing undecidability

Mapping reductions for showing NP-completeness: the SAT to Clique reduction.

BDDs for simple Boolean functions.

BDDs as minimal DFA

BDDs used to express Logic

BDDs to synthesize circuits using multiplexors

How many Boolean functions over N inputs

How to use a 4-to-1 mux to implement all possible 2-input Boolean functions

Variable ordering

What it means for a problem to be NP-complete

How Regular, DCFL, CFL, CSL, NPC, Decidable, RE, non-RE are contained.

Show that $EQ_{TM}$ is not RE.

Proof: Build two machines $M_1$ which always rejects and $M_2$ which accepts any $x$ so long as $M$ accepts $w$. Then we have achieved $A_{TM} \leq_m EQ_{TM}$. This means $A_{TM} \leq_m EQ_{TM}$. Thus, $EQ_{TM}$ can’t be RE (else we will have an enumerator for $A_{TM}$.