CS 3100 – Models of Computation – Fall 2011

FINAL EXAM – CLOSED BOOK – 60 points – 100 minutes recommended

Your Name: **Your UNID:**

1. **(2 points)** State the Schröder-Bernstein (S-B) theorem in one sentence.

   For any two sets A and B, there is a one-one, onto, and total map f going from A to B exactly when there are two other one-one, into, and total maps $f : A \rightarrow B$ and $g : B \rightarrow A$.

   Apply the S-B theorem to prove that the set of all Rational numbers has the same cardinality as the set of all *Even* numbers, by presenting the mappings requested below. Rationals are those numbers that can be expressed in the form x/y:

2. **(3 points)** Present the 1-1, total, and into mapping going from Rationals to Even numbers as a Lambda function. Show, by the way of examples, what `3/4` and `4/3` end up mapping to.

   $\lambda(x, y) : 2 \times (2^x \times 3^y)$ would do. Here, x, y are the rational's numerator and denominator. Example: $3/4$ maps to $16 \times 81 = 1296$.

3. **(3 points)** Present the 1-1, total, and into mapping going from Even numbers to Rational numbers as a Lambda function. Show, by the way of examples, what `0` and `2` end up mapping to.

   Use $\lambda x : 1/(x + 2)$. The $+2$ in the denominator avoids division by 0.

   Solutions such as $\lambda x : x/1$ are fine because those are rationals. Then there is no division by 0 happening.

4. **(6 pts)** Prove that the following language is recursively enumerable. Your proof must be a precise English argument presented as bulletted steps that shows how to enumerate the contents of this language. Your steps must clarify exactly when you would list a grammar in this listing.

$$G_{amb} = \{\langle G \rangle \mid G \textit{ is an ambiguous } CFG\}$$

   - Enumerate all possible grammars G and strings w in Σ^* in a dovetailed manner.
   - Generate all possible parse trees for w with respect to G, if parseable.
   - If some G admits one string that has more than one parse, output G into the listing.
   - This is an enumerator for G_{amb}.

5. **(2 pts)** Why does it seem intuitively hard to recursively enumerate this language? (Two short sentences.)

$$G_{unamb} = \{\langle G \rangle \mid G \textit{ is an unambiguous } CFG\}$$

   To enumerate *unambiguous* grammars, you have to try *all* strings and conclude that for none, we have an ambiguous parse. This won't terminate.

6. **(2 pts)** What established result would be contradicted if G_{unamb} were also to be RE? (Two short sentences.)

   If both the languages are RE, the language becomes decidable, which is a contradiction.

7. **(10 pts)** Prove by diagonalization that the set of languages over $\Sigma = \{0\}$ has cardinality $\aleph_1$. Clearly write down the steps, showing how an attempted listing looks like, and how the diagonalization argument is applied.

   Students must show the claimed matrix of strings and languages

```
    @ 0 00 000 0000 00000 ... strings

  0 0 1 0  1

  1 1 0 1  1 ..
  2
  3
  4
...
languages
```

   Then they claim that the diagonal complemented is a language not in the listing.

8. Someone claims that the following is a mapping reduction from A_{TM} to CFG_{TM}, where the latter is the language of all TMs (TM codes, actually) that have a context-free language. They explain that given M and w, the mapping reduction produces M' described below.

```
M'(x) {
  if x is of the form "ww" for some string w in Sigma-star, THEN goto accept_M' ;
  Run M on w ;
  If M accepts w, goto accept_M' ;
  If M rejects w, goto reject_M' ; }
```

   (a) **(3 pts)** Show that if M accepts w, then the language of Turing machine M' is context-free. (2-3 clear sentences.)

   The language of the machine in fact is "all strings", or regular, if M accepts w. (Reasons were discussed in class; they must give reasons.)

   (b) **(3 pts)** Show that if M does not accept w, then the language of Turing machine M' is not context-free. (2-3 clear sentences.)

   If M does not accept w, we are left with ww strings, which are not going to be a CFL. (Reasons were discussed in class; they must give reasons.)

   (c) **(3 pts)** How does this construction give you a decider for A_{TM}, given a decider for CFG_{TM}? Draw the "boxes within boxes" picture of how you can build an A_{TM} decider, given a CFG_{TM} decider. You can assume that there is a little "box" that, given M and w, builds M' for you (label this box M' builder).

You can use the M' builder and generate this machine out of M and w. Next, feed it to the claimed CFG decider. The results of this CFG decider form the accept/reject outputs of the bigger box which is now the A_{TM} decider.

9. **(7 pts)** Write a clear proof that H_{TM}, the decider for halting Turing machines, cannot exist. Write 5-6 precise and short steps in English + math notation.

   As discussed in class. How the $D(D)$ results in a contradiction.

10. **(3 pts)** A student writes a DFA minimizer and it loops infinitely. She claims that this is to be expected, given that the Halting problem is undecidable. How would you refute this student's argument (in 2-3 short, and technically precise steps – no glib or "funny" answers expected).

    This is one particular program whose termination can be proven. That does not contradict the halting problem. The halting problem is a much more general statement about all programs - not a particular program. So in this case, there is a coding bug that needs to be corrected, and then the minimizer will halt on all inputs.

11. You are asked to construct a BDD for the Boolean function $majority(x, y, z)$ over variables x, y, z. This function outputs a 1 (or "true") exactly when a majority of x, y, z are 1s (*i.e.*, there are more 1 bits than 0 bits, in any binary setting of these variables).

    (a) **(3 pts)** Does it matter which variable ordering you pick? Why or why not? (2-3 clear sentences).
    No it does not matter. We are counting the bits, and no bit affects the overall decision more than the others. So the BDD size will be the same no matter how you build the BDD.

    (b) **(7 pts)** With respect to a "good" variable order, draw the BDD for this function.
    Must draw a BDD where no redundant decodings are shown.

    ```
    Here is what the solution will look like.


        x
    1 /  \ 0
    y1     y2

    z1     z2

    1     0

    Now
    y1 on a 1 goes to 1. y1 on a 0 goes to z2.
    y2 on a 1 goes to z1. on a 0, it goes to 0.
    z1 on a 1 to 1; else 0
    z2 on a 1 to 1; else 0
    ```

    (c) **(3 pts)** How are BDDs related to DFA? Take the majority function as an example; how does the BDD you drew for it relate to the language of some DFA? Describe that language mathematically.
    This will define the minimal DFA for the language

    ```
     { xyz  | majority(x,y,z) }
    ```

Extra Pts. Toward Midterms (20 mins recommended; 20 pts max)

These points will be turned into compensating points toward Midterm-1 and Midterm-2 using a formula TBD. You are welcome to work on these problems below, or simply ignore them and work on the previous problems which fetch you points for the finals.

1. **(10 pts)** Draw a DFA for the complement of the language denoted by the regular expression `(0101)*`.
   Draw a min DFA for (0101)* and complement the states.

```
 --> s0 -0--> s1 -- 1 --> s2 --0--> s3 --1--> s4  --0-->> back to s1
     ACC      ACC          ACC        ACC         REJ

 all other moves go to a BH

Self-loop on BH (optional to show)
```

2. **(10 pts)** Write a CFG for $\{ab \mid length(a) = length(b),\ and\ a \neq b, where\ a, b \in \{0,1\}^*\}$
   This is almost the complement of the language

   `{w w | ..}`

   The only case missing from the complement of

   `{w w | ..}`

   is the odd-length case.
   Recall the construction

```
--------x--- --------y---

where x is a 1 and y is a 0
or    x is a 0 and y is a 1
```

   So here we go

```
S -> X  Y  |  Y  X

X -> D X D | 0

Y -> D Y D | 1

D -> 0 | 1
```

   I saw that most of the other attempted answers accidentally included strings of the form `ww`.
   I also saw many attempts where not all the strings were included.