CS103 Practice Midterm Exam

This midterm exam is open-book, open-note, open-computer, but closed-network. This means that if you want to have your laptop with you when you take the exam, that's perfectly fine, but you **must not** use a network connection. You should only use your computer to look at notes you've downloaded in advance. Although you may use laptops, you **must** hand-write all of your solutions on this physical copy of the exam. No electronic submissions will be considered without prior consent of the course staff.

Normally, we would provide space on the exam for you to write your answers, but in the interest of saving paper we've eliminated most whitespace from this practice exam.

You have three hours to complete this midterm. There are 180 total points, and this midterm is worth 15% of your total grade in this course. You may find it useful to read through all the questions to get a sense of what this midterm contains. As a rough sense of the difficulty of each question, there is one point on this exam per minute of testing time.

Question		**Points**	**Grader**
(1) Translating into Logic	(20)	/ 20	
(2) Cantor's Theorem Revisited	(25)	/ 25	
(3) Tournament Victory Chains	(45)	/ 45	
(4) Composing Relations	(45)	/ 45	
(5) Inherent Complexity	(45)	/ 45	
	(180)	**/ 180**	

Good luck!

Problem One: Translating into Logic (20 points total)

In each of the following, you will be given a list of first-order predicates and functions along with an English sentence. In each case, write a statement in first-order logic that expresses the indicated sentence. The statement you write should can use any first-order construct (equality, connectives, quantifiers, etc.), but you must only use the predicates and functions provided.

As an example, if you were given just the predicates *Integer(x)*, which returns if x is an integer, and the function *Plus(x, y)*, which returns $x + y$, you could write the statement "there is some even integer" as

$$\exists n.\ \exists k.\ (Integer(n) \land Integer(k) \land Plus(k, k) = n)$$

since this asserts that some integer n is equal to $2k$ for integer k. However, you could not write

$$\exists n.\ (Integer(n) \land Even(n))$$

because there is no *Even* predicate. The point of this question is to get you to think how to express certain concepts in first-order logic given a limited set of predicates, so feel free to write any formula you'd like as long as you don't invent your own predicates or functions.

(i) It's Turtles All The Way Down (5 Points)

Given the predicates

Turtle(t), which says that t is a turtle, and

Beneath(s, t), which says that s is beneath t,

write a statement in first-order logic that says "below every turtle is some other turtle."

(ii) The Well-Ordering Principle (15 Points)

Given the predicates

Set(S), which says that S is a set,

$x \in S$, which says that x is an element of S, and

$x < y$, which says that x is less than y,

along with the constant $\mathbb{N}$, which represents the set of all natural numbers, write a statement in first-order logic that says "every nonempty set of natural numbers has a least element." A least element of a set is an element of that set that is strictly smaller than all other elements of that set.

Problem Two: Cantor's Theorem Revisited (25 points)

Cantor's theorem states that $|S| < |\wp(S)|$ for all sets S. Among other things, this theorem says that $|\mathbb{N}| < |\wp(\mathbb{N})|$, meaning that there are strictly more sets of natural numbers than natural numbers.

Let's denote by $\wp_{<\omega}(S)$ the set of all *finite* subsets of set S. For example, $\{1, 2, 3\} \in \wp_{<\omega}(\mathbb{N})$, but $\mathbb{N} \notin \wp_{<\omega}(\mathbb{N})$ because $\mathbb{N}$ is infinite. Amazingly, it turns out that there are the same number of natural numbers as there are finite subsets of natural numbers; that is, $|\mathbb{N}| = |\wp_{<\omega}(\mathbb{N})|$.

This result seems surprising, and so it might be tempting to try to disprove it. Of course, since the result actually is true, it cannot be disproven. But since when has that stopped people from trying?

(i) Injectivity, but not Surjectivity (10 Points)

Below is an **incorrect** proof that suggests that $|\mathbb{N}| \neq |\wp_{<\omega}(\mathbb{N})|$. This proof contains a logical error that renders it invalid. Identify the flaw in the reasoning. You do not need to give an explicit counterexample; just state what logical error is being made.

Theorem: $|\mathbb{N}| \neq |\wp_{<\omega}(\mathbb{N})|$.

Proof: Consider the function $f : \mathbb{N} \to \wp_{<\omega}(\mathbb{N})$ defined as follows: for any $n \in \mathbb{N}$, let $f(n) = \{n\}$. To see that this is a valid function from $\mathbb{N}$ to $\wp_{<\omega}(\mathbb{N})$, note that for any $n \in \mathbb{N}$, we know that $\{n\} \subseteq \mathbb{N}$ and that $\{n\}$ is finite. Thus $f(n) \in \wp_{<\omega}(\mathbb{N})$.

We claim that f is injective. To see this, consider any $n_0 \in \mathbb{N}$ and $n_1 \in \mathbb{N}$ such that $f(n_0) = f(n_1)$. We will prove that $n_0 = n_1$. Since $f(n_0) = f(n_1)$, we know that $\{n_0\} = \{n_1\}$. Since two sets are equal iff they contain the same elements, this means that $n_0 = n_1$, as required.

However, f is not surjective. To see this, note that $\{0, 1\}$ is a finite subset of $\mathbb{N}$, so $\{0, 1\} \in \wp_{<\omega}(\mathbb{N})$. However, there is no $n \in \mathbb{N}$ such that $f(n) = \{0, 1\}$, so f is not surjective. Since f is not surjective, it is not bijective. Thus $|\mathbb{N}| \neq |\wp_{<\omega}(\mathbb{N})|$. ■

(ii) The Diagonal Argument Revisited **(15 Points)**

In the proof of Cantor's theorem, we used a diagonal argument to show that $|S| \neq |\wp(S)|$. Below is an **incorrect** proof that tries to use diagonalization to show that $|\mathbb{N}| \neq |\wp_{<\omega}(\mathbb{N})|$. As with part (i), this proof contains a logical error that renders it invalid. Identify the flaw in the reasoning. You do not need to give an explicit counterexample; just state what logical error is being made.

Theorem: $|\mathbb{N}| \neq |\wp_{<\omega}(\mathbb{N})|$.

Proof: By contradiction; assume that $|\mathbb{N}| = |\wp_{<\omega}(\mathbb{N})|$, so there is a bijection $f : \mathbb{N} \to \wp_{<\omega}(\mathbb{N})$. Consider the set $D = \{ n \in \mathbb{N} \mid n \notin f(n) \}$. Since f is a bijection, it is surjective, so there must be some $d \in \mathbb{N}$ such that $f(d) = D$. Now, either $d \in D$, or $d \notin D$. We consider these cases separately:

Case 1: $d \in D$. By our definition of D, this means that $d \notin f(d)$. However, we know that $f(d) = D$, so this means that $d \notin D$, contradicting the fact that $d \in D$.

Case 2: $d \notin D$. By our definition of D, this means that $d \in f(d)$. However, we know that $f(d) = D$, so this means that $d \in D$, contradicting the fact that $d \notin D$.

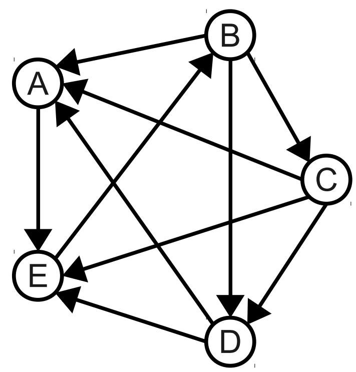
In either case we reach a contradiction, so our assumption must have been wrong. Thus $|\mathbb{N}| \neq |\wp_{<\omega}(\mathbb{N})|$. ■

Problem Three: Tournament Victory Chains (45 points)

Consider any tournament graph for $n > 0$ players. Let's define a **victory chain** as a sequence of players where

- Every player is in the sequence,
- No player is repeated in the sequence, and
- For every player in the sequence (except the last), that player beats the player that comes after her.

For example, consider this tournament graph:



One possible victory chain is **A → E → B → C → D**, since A beat E, E beat B, B beat C, and C beat D. Another possible victory chain is **C → D → E → B → A**, since C beat D, D beat E, E beat B, and B beat A. However, **B → A → E → C → D** is not a victory chain, since E lost to C. **B → A → E → B → C → D** is not a victory chain because B is repeated in the sequence, and **D → A → E → B** is not a victory chain because C is nowhere in the chain.

Prove that any tournament with $n > 0$ players has at least one victory chain. Although you have proven in the problem set that all tournament graphs have at least one tournament winner, you do not need to use that result in this proof. That said, you still might find induction useful.

Problem Four: Composing Relations (45 Points)

Suppose that R and S are binary relations over some set A. The **composition of *R* and *S***, denoted $S \circ R$, is the relation defined as follows:

$$\{ (x, y) \in A \times A \mid \text{there is some } z \in A \text{ such that } xRz \text{ and } zSy \}$$

Restated in English, $x(S \circ R)y$ iff there is some z such that xRz and zSy.

One interesting special case to consider is the composition of a relation with itself. Given a binary relation R, the relation $R \circ R$ is defined as follows: $x(R \circ R)y$ iff there is some z such that xRz and zRy. We use the notation R^2 to denote $R \circ R$.

Given a relation R, what is the connection between R and R^2? Are these relations always different? Are they always the same? In this problem, you will explore this question.

(i) Equivalence Relations (15 Points)

Prove or disprove: If R is an equivalence relation over a set A, then $R = R^2$. Recall that $R = R^2$ when for every $x \in A$ and $y \in A$, we have that xRy iff xR^2y.

(ii) Partial Orders (15 Points)

Prove or disprove: If R is an partial order over a set A, then $R = R^2$.

(iii) The Converse? (15 Points)

Prove or disprove: If R is a relation over a set A and $R = R^2$, then R is an equivalence relation or a partial order.

Problem Five: Inherent Complexity (45 points total)

As mentioned in lecture, regular languages are precisely the languages that can be accepted by some DFA. However, within the set of regular languages, some languages are more complicated than others.

For any natural number n, consider the language $L_n = \{ w \mid w \text{ has length at least } n \}$ over the alphabet { `0`, `1` }. For example, $L_0 = \Sigma^*$, since every string has length at least 0. The language L_1 is the set {`0`, `1`, `00`, `01`, `10`, `11`, … }, and L_5 = { `00000`, `00001`, `00010`, … }.

Prove that for any $n \in \mathbb{N}$, that there does not exist a DFA D satisfying the following two properties:

- $\mathscr{L}(D) = L_n$ (that is, D is a DFA for the language L_n), and
- D has strictly fewer than $n + 1$ states.

This shows that some regular languages are inherently complicated, since any DFA for them must be at least some minimum size.

(Hint: Use the pigeonhole principle. If the DFA has fewer than $n + 1$ states, then what would happen if you ran the DFA on $n + 1$ specially-chosen strings?)