CSC238:Discrete Mathematics for Computer Science – Spring 2003

Assignment 4: Predicate Logic and Regular Expressions

Due: 10 a.m., Thurs., April 10

This assignment is worth 10 percent of the total marks for this course.

This assignment provides basic exercises in predicate logic, regular expressions, and finite state automata. We will mark only a (secret) subset of the questions below. As always in this course, your proofs will be marked for correctness, along with brevity and clarity.

Your answers can be handwritten. Use standard 8.5 by 11 inch paper. Please staple all the sheets together, and hand them to your tutor at the **beginning** of the tutorial on the due date. If you cannot make it to that tutorial, then leave your assignment at your instructor's office (Pratt, Room 283) before 10am on the due date. Since the tutors will be discussing the solutions in the tutorial immediately after your assigments are due, we will not accept late assignments (the course homepage describes what you should do in case of medical or other emergencies which prevent you from completing an assignment on time).

1. For each of the following assertions, state whether it is true or false, and justify your answer. You can use any means of justification you like, but if you use the logical equivalences stated in the course notes then clearly specify which ones. Here $A(x)$ and $B(x)$ are unary predicates in the first order language $\mathcal{L}$.

   (a) $\forall x(A(x) \to B(x))$ logically implies $\exists x(A(x) \land B(x))$.

   (b) $\exists x(A(x) \to \neg B(x))$ is logically equivalent to $\neg\forall x(A(x) \land B(x))$.

   (c) $\exists x A(x) \land \exists x \neg A(x)$ is logically equivalent to $\exists x(A(x) \land \neg A(x))$.

   (d) $\exists x A(x) \lor \exists x \neg A(x)$ is logically equivalent to $\exists x(A(x) \lor \neg A(x))$.

2. Convert the following first order formula into a logically equivalent formula in Prenex Normal Form (PNF) which only uses the connectives $\neg$, $\land$ and $\lor$ (along with the quantifiers $\exists$ and $\forall$, of course). In particular, **do not use** the connective $\to$.

$$(\forall x \neg T(z,x) \to S(x,y,z)) \ \land \ (\neg\exists y \forall z R(y,z,x) \to \exists x U(x)).$$

   Show the steps through which you obtained your PNF formula.

3. Consider a first order language $\mathcal{L}$ with predicates $S(s,n,a)$, $C(c,n)$, $E(s,c,y,m)$ and $T(n,c,y)$, and the equality relation $\approx$. Consider a structure for $\mathcal{L}$ which consists of the following relations corresponding to the predicates S, C, E, and T, respectively:

   - $Student(s,n,a)$ – each student corresponds to a unique triple (s,n,a), where s is the unique student number, n the student's name, and a is the student's address (all in the form of character strings).

- $Course(c, n)$ – each course corresponds to a unique pair (c, n) where c is the unique course identifier (e.g. "CSC238") and n is the course title (e.g. "Discrete Mathematics for Computer Science").
- $Enrolled(s, c, y, m)$ – a quadruple (s, c, y, m) indicates that the student with student number s was enrolled in the course c in the year y and obtained the mark m. If the course is currently being taken by the student, then the mark is recorded as an "I" for incomplete.
- $Teaches(n, c, y)$ – a triple (n, c, y) indicates that a professor named n taught the course named c in the year y.

The domain D for this structure is the set of strings that form entries in this database. For example, these strings include all the student numbers, names, addresses, course names, course identifiers, years, and so on. We will also use these same strings as constants in the first order language, so a formula such as $(n \approx \text{“}Jepson\text{”})$ is true only when the valuation σ is such that $\sigma(n)$ is the string "Jepson" in the domain D.

For a more complex example, consider the first order formula

$$\exists y E(s, \text{“}CSC238\text{”}, y, m) \wedge ((m \approx \text{“}A\text{”}) \vee (m \approx \text{“}B\text{”})).$$

Since s and m are the only free variables, we will refer to this formula as simply $F(s, m)$. Given the above structure, $F(s, m)$ is true for any valuation σ for which the pair $(\sigma(s), \sigma(m))$ consists of the student number $\sigma(s)$ and the mark $\sigma(m)$ such that the student with that number took CSC238 in some year and got a mark of either A or B.

In this manner it is natural to associate the first order formula $F(s, m)$ with the set of all such student number and mark pairs $(\sigma(s), \sigma(m))$ which make $F(s, m)$ true for some valuation σ. That is, the formula $F(s, m)$ determines a relation in $D \times D$ consisting of all such pairs of student numbers and marks. This is equivalent to thinking of the first order formula $F(s, m)$ as expressing a database query of finding all the student numbers and marks for all the students who have ever taken CSC238 and received a mark of either A or B in it.

In a similar fashion, write down a first order formulas expressing each of the following queries:

(a) Find the name and address of every student who took CSC238 in 2001.

(b) Find the names of all the courses ever taught by Cook.

(c) Find the names of all the courses taught *only* by Cook.

(d) Find the names of every student who has received an "A" or an "I" in every course he/she has taken.

(e) Find the names of every course that has neither been taken, nor is still incomplete, by any student who has ever enrolled in CSC238.

4. Consider the alphabet $\Sigma = \{0, 1\}$ and the language

$$L = \{x \in \Sigma^* : 01 \text{ and } 10 \text{ are both substrings of } x\}.$$

For example, 0001110010101 and 11101 are in L. Note that $010 \in L$ since 01 and 10 are both substrings of 010, even though these substrings overlap on the character 1. Construct a regular expression that denotes L, and prove that it is correct.

5. For each of the following languages over the alphabet $\Sigma = \{0,\ 1\}$, construct a regular expression that denotes it and construct a DFSA that accepts it. You do not need to provide proofs that your constructions are correct.

   (a) $L = \{x \in \Sigma^* : x \neq \epsilon$, and the first and last symbols of x are different$\}$,

   (b) $L = \{x \in \Sigma^* : x = 0^n 1^m$ with $n, m \geq 0$ and $n + m$ is odd$\}$,

   (c) $L = \{x \in \Sigma^* :$ every 0 in x (if any) is immediately followed by a 1 $\}$.

6. Consider the alphabet $\Sigma = \{0, 1\}$, and language

$$L_3 = \{x \in \Sigma^* : x = y1z \text{ with } y, z \in \Sigma^* \text{ and } z \text{ has length } |z| = 2\}.$$

That is, L_3 is the language of binary strings of length at least 3 for which the third last character is 1.

   (a) Write a regular expression which denotes L_3.

   (b) Construct a NFSA with 4 states which accepts L_3.

   (c) Construct a DFSA with 8 states which accepts L_3. (Hint: Label the eight nodes 000, 001, ..., 111, and define the DFSA such that the last three characters b_{k-2}, b_{k-1}, b_k processed by the DFSA correspond to the label abc of the state. Note that in order to start processing input you need to determine the corresponding state for inputs of length less than 3. To do this, treat any leading zeros in the state label abc as potentially a missing item. For example, the start state should be the one labelled 000, and if the first character is a 1 then the DFSA should then move to state 001.)

   (d) Consider generalizing the definition of L_3 by defining L_k to be the language in Σ^* such that the k^{th} bit from the right end is 1. Show the form of a $k + 1$ state NFSA which accepts L_k for each $k \geq 1$. How many states are required for a DFSA which accepts L_k (see the hint in part c above)? Explain (but you do not need to prove anything here).