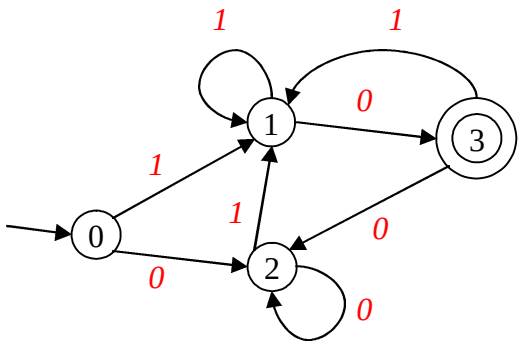
Unit C5: Languages/Automata, 11/7/03

Sample solutions to these exercises will be posted in the afternoon of the due date. But do these problems without seeing the solutions. Being able to understand provided solutions is *completely* different from being able to come up with a solution. Even an incomplete or incorrect answer of your own is better than understanding someone else' answer.

Exercise 1: Binary Number Analyzer

Finite-State Automata (FSA) are an extremely useful tool to analyze simple patterns in input strings. For example, the following FSA is designed to accept a certain set of binary numbers.



The FSA starts at State 0 (circled 0) and changes its state as the input symbols (*0* and *1* in italic) is consumed from left to right. For example, if the input is *01*, the machine starts from State 0, moves to State 2 following the transition for the first input symbol *0*, and then makes one more move on the second symbol *1* to State 1. The FSA accepts an input if it is in State 3 (a double circled state) after consuming all the input symbols. Thus, the input *01* will be rejected, because the FSA ends up in State 1 (not a double circled state).

A. List all the input strings of length 1 through 4 that will be accepted by the above FSA.
   Note: Include all sequences of 0's and 1's, e.g., those with leading 0's, e.g., 010.

B. Describe (in English) the properties shared by all the acceptable input strings (not limited to those listed in Question A.).
   Hint: Focus on how the acceptable inputs end.

C. If these binary numbers are converted to the corresponding decimal numbers, the set of the decimal numbers can be represented as $i \times n + j$ for any natural number n (with an appropriate fixed numbers i and j). Find the correct i and j.
   Hint: The set of even and odd numbers can be represented as $2n + 0$ and $2n + 1$, respectively.

D. Give a *single* regular expression that would specify exactly the set of acceptable input strings.
   Note 1: A single regular expression can specify a set of strings. Do not give multiple regular expressions.

Note 2: Strictly speaking, a regular expression consists of *sets* and operations on sets. Thus, the regular expression to represent a sequence of '0' and '1' must be written {0}{1}. However, when we deal with singletons, the expression can easily get unreadable. For this reason, we often abbreviate singletons such as {1} as 1, and consider 01 as a regular expression.

E. If States 0 and 2 are merged, would the resulting FSA still accept the same language? Explain.

Exercise 2: English Spelling

In English, it is almost always the case that letter '*q*' is followed by letter '*u*' and then a vowel letter, i.e., '*a*', '*e*', '*i*', '*o*', '*u*'. In order to represent this phenomenon, let us first introduce the following sets:

- $A = \{a, b, c, ..., z\}$
- $Q = \{q\}$
- $U = \{u\}$
- $V = \{a, e, i, o, u\}$
- $X = A - Q$

Then, the set of all the strings that abide by the above-mentioned rule can be represented by the following regular expression: $((QUV) \mid X^*)^*$ For example, *esquire*, *queue*, and *quay* are in the set, but *qantas* is not.

Give a Finite-State Automaton that would accept exactly the set of strings characterized by the given regular expression.

Note: Correctly indicate the start and accepting states, as discussed in class.

Exercise 3: Musical Code Sequence

Consider the set of musical codes, *Code* = {A, B, C, D, E, F, G}. Suppose that an amateur musician made the following discovery about the contemporary Aztec music.

1. F is always followed by C.
2. D and E always occur consecutively, in that order.
3. A and B never appear.
4. Silence is an acceptable (empty) code sequence.

According to this analysis, we notice the following cases:

- Acceptable sequences: FC, DEFC, GFCG~~F~~DEG, ⟨empty code sequence⟩
- Unacceptable sequences: CF, DCFE, AFCDE

A. Give a regular expression that would accept exactly the set of acceptable code sequences.

Hint: Identify the three components of regular expression: sequence, alternative, and repetition. For example, you can represent Conditions 1 and 2 as alternative sequences. Then, you should be able to repeat valid code segments arbitrarily long.

B. Create a Finite-State Automata that would accept exactly the set of acceptable code sequences.
Note: Make sure that the transition of your FSA is a function. That is, when a state and an input are specified, there must be a unique next state.
Hint: Create a FSA corresponding to the regular expression in Question A.

<End>