CMSC210 Module C Comprehensive Exercises

Due at the beginning of the class meeting on 11/14/03

1. String Theory

Weird things do happen. Philosophers, Philanthropists, and Physicists may all have their own ideas about how to explain such things. For example, some Physicists believe that weird things can happen because we live in a modest 11-dimensional world as predicted by "String Theory."

Note: If you are slightly interested in this theory, visit the official (?) web site: http://superstringtheory.com/. Or, you can also borrow from the instructor a video recording of a recent PBS program NOVA: "The Elegant Universe." But as in some previous exercises, that will not likely to be helpful to do this exercise.

In a class, we noted that Boolean Algebra could be used to visualize a multi-dimensional world. Then, maybe we can explore a bit of String Theory using Boolean Algebra, formally represented as **Bool** = (B, $\cap$, $\cup$, ', *Top*, *Bottom*). Here, we use the operation/function symbols from Set Theory and two constants with names that correspond to their positions in their Hasse diagrams.

A. (i) Give the cardinality of B that would represent an 11-dimensional world. (ii) Formally define B (for the same 11-dimensional world) using a method of your choice [not wise to use the list notation, though].

B. Draw a Hasse diagram of the poset *equivalent to* the Boolean Algebra discussed in Question A.

    Note: Do not even attempt to draw the entire Hasse diagram, which will be HUGE. Limit your drawing to the elements directly connected to *Top* or *Bottom*, and then draw a blob, dotted lines, and/or some sort of indication how the remaining elements could be placed.

C. Consider the Boolean Algebra discussed in Question A and schematically represented in Question B. (i) If you interpret this Boolean Algebra in the context of computer architecture, what can you say? (ii) Would a typical 32-bit computer architecture be capable of computing logical *and* and *or* of the values in B by applying one of the operations (corresponding to $\cap$, $\cup$, and ') just once (i.e., in a single CPU cycle)? Explain.

D. Consider the poset *equivalent to* the Boolean Algebra discussed in Question A and schematically represented in Question B. Then, imagine the following scenario: due to some change in the universe, the poset (as viewed through its Hasse diagram) suddenly lost the *Top* element and all of its connections. Let's call the structure after the change **Crunch**. (i) Would **Crunch** still correspond to a Boolean Algebra? (ii) Is **Crunch** still a poset?

E. (i) Discuss why not all the phenomena around us can be represented as a Boolean Algebra. (ii) Then, also discuss the implications of being able to represent an 11-dimensional world as a Boolean Algebra, in connection to your discussion of (i).

2. Program Flow Analysis

When a compiler translates a high-level language (e.g., Java, C++) into the target machine language (e.g., for an Intel or SPARC processor), it analyzes the given program with respect to many properties. One aspect involves how program statements (commands) are executed and how variables (internal storage of numbers and other things) are assigned values.
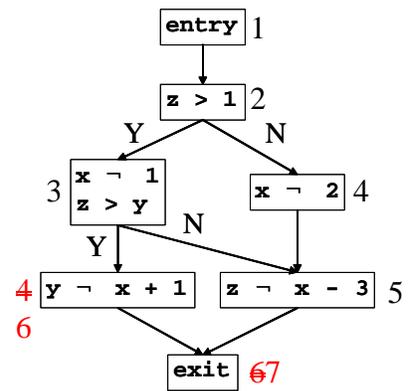
The diagram at right is an example of analyzing a small portion of a program. Boxes represent states (numbered for referencing purposes) and arrows represent state transitions. For example, State 2 occurs after State 1. 'Y'/'N' indicates a conditional branch based on value comparison (e.g., Z > 1), but this point is not relevant in this exercise.

A. Formally represent the diagram as a digraph, focusing on the states and transitions (do not even think about the content in each box and 'Y'/'N' indication), and call it **Flow**. All the structure components must also be formally defined.

B. Would it be possible to modify the diagram so that it can be represented as a "tree," while program flow represented by the modified tree diagram would remain the same as the original diagram? Explain.

   Hint: Would splitting State 6 into two separate states, say, States 6a and 6b, change the ability to analyze the same program?

C. We can conclude that the program segment represented by the flow diagram is guaranteed to terminate after some finite steps of transition. What property of a graph can guarantee program termination? Explain.

   Note: You must be specific about the distinction between digraph and undirected graph.

D. Suppose that someone added one more state to the diagram (where and how it is added is not disclosed to you). (i) Discuss whether the modified diagram has a corresponding Boolean Algebra. (ii) Discuss whether the digraph corresponding to the modified diagram is a poset.

E. Discuss whether the (original) diagram can be represented as a Finite-State Automata (FSA). If yes, what would be the *structure* that is specified by the FSA (when the FSA is seen as a kind of logic)? If no, explain why not.

F. Suppose that you are given a diagram of another program section and found that the corresponding graph representation is not "connected." What would be the implication of this finding?

3. Advanced Web Search

There is no doubt about the usefulness of web search engines, e.g., Google. However, in certain cases, you might want to use more advanced search criteria such as the following:

Search Criteria
1. Find all and only the documents that contain *hot* and *dog* in that order, but not necessarily next to each other.
2. Find all and only the documents where every occurrence of *cat* is followed by *woman*. However, *cat* is not required and *woman* can appear freely in addition to the required positions.
3. Find all and only the documents that contain even number (including zero) of any of the following in any order: *no*, *non*, *nao*, and *nyet*.

Let us limit the discussion to the set of printable ASCII characters and denote it as Σ. That is, Σ = {' ' (space), '!', ..., 0, 1, ..., 9, ..., A, B, ..., Z, ..., a, b, ..., z, ... '~'}, where special symbols are single-quoted.

A. Let's hope that Google would eventually accept regular expressions as their inputs. Give a regular expression corresponding to each one of **Search Criteria**.

   Note: There are variants of regular expressions. Use the notation discussed in class slides.

B. For each of the regular expressions you obtained in Question A, define an equivalent Finite-State Automaton (FSA).

   Note: Use the diagram notation shown in class slides, including the indication of the start state, input symbols, transitions (directional), and accepting/final states.

C. (i) Explain what kind of structure would a regular expression (viewed as a kind of logic) specify. (ii) Explain what kind of structure would a FSA (also viewed as a kind of logic) specify. (iii) Identify the learning goal that is most relevant to connecting your answers to (i) and (ii) above. Explain.

   Hint: Review the definition of "language."

D. Although we viewed FSA as a kind of logic (i.e., a means to specify certain structure), it is also possible to view them as a structure. In fact, the formal definition of a FSA take the form of a structure, i.e., an *n*-tuple with sets and functions. In this view of FSA's, what would be the logic part that would specify a certain collection of FSA's?