Contest 3 Review

George Tang and Neal Bayya
Compiled by Anna Xu

February 2018

1 Boolean Algebra

Boolean algebra is essentially algebra done with 1 and 0, true and false. There are operations, AND, OR, and XOR, and a specific order of operations to remember: NOT, AND, XOR, OR. ACSL will mostly ask you to simplify a boolean expression or to find the solutions to an expression such that it returns True or False.

2 Boolean Identities

You should know these identities:

1.A+B = B+A
2.A$\bullet B = B \bullet A$
3.$(A + B) + C = A + (B + C)$
4.$(A \bullet B) \bullet C = A \bullet (B \bullet C)$
5.$A + (B \bullet C) = (A + B) \bullet (A + C)$
6.$A \bullet (B + C) = AB + AC$
7.$\overline{A + B} = \overline{A} \bullet \overline{B}$
8.$\overline{A \bullet B} = \overline{A} + \overline{B}$
9.$A + 1 = 1$
10.$A \bullet 1 = A$
11.$A + 0 = A$
12.$A \bullet 0 = 0$
13.$A + A = A$
14.$A \bullet A = A$
15.$A + \overline{A} = 1$
16.$A \bullet \overline{A} = 0$
17.$\overline{\overline{A}} = A$
18.$A \oplus B = A\overline{B} + \overline{A}B$
19.$\overline{A \oplus B} = \overline{A} \oplus B = A \oplus \overline{B}$
20.$A \bullet (A + B) = A$

3 Data Structures

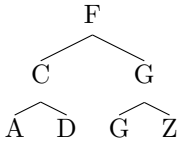
The three data structures used in ACSL are stacks, queues, and trees. Stacks are FIFO whereas queues are FILO. They both have commands that follow the format PUSH(X) and POP(X).

Trees used in ACSL are binary trees and either involve construction or vocabulary. For construction, a word will be given and a corresponding tree must be built. Each letter is added to the tree and its location is found by taking the current node and going left if it is alphabetically equal or earlier than the node, or otherwise going right until an empty slot is found.

3.1 Min and Max Heaps

A min heap is where the root is the smallest value of the tree and each node is smaller than its children.A max heap is where the root is the biggest value of the tree and each node is bigger than its children.

Example Tree:



The necessary vocabulary includes:

1. Depth/Height: Distance from deepest node to root Ex: 2
2. Leaf Nodes: Nodes that don't have children Ex: 4
3. External Nodes (only for binary trees): Places where new nodes could be added (Not drawn) Ex: 8
4. Internal Path Length: Sum of depths of all nodes Ex: 8
5. External Path Length: Sum of depths of all external nodes. Ex: 24

4 Regex

ACSL Regex is a method to match strings containing either 1s or 0s. The three operations used are concatenation (ab), union (aUb), and closure (a*). They mean a followed by b, a or b, and a 0 or more times respectively. Graphically, concatenations are represented by lines, unions by branches, and closures by loops going from one node to itself. An FSA ends when it reaches a double circle. Typical problems in this category will include: translate an FSA to/from a regular expression; simplify an FSA or regular expression (possibly to minimize number of states or transitions); create an FSA to accept certain types of strings.

5 Regex Identities

1. (a*)* = a
2. aa* = a*a

3. aa*U = a*
4. a(b U c) = ab U ac
5. a(ba)* = (ab)*a
6. (a U b)* = (a* U b*)*
7. (a U b)* = (a*b*)*
8. (a U b)* = a*(ba*)*

6 Practice

1. Simplify $AB + ABC + ABCD + ABCDE + CDE$
2. Simplify $\overline{A} \bullet \overline{B} + \overline{A + \overline{C}} + \overline{A} \bullet B \bullet C$
3. Simplify $(A + B) \oplus AB \bullet AC + \overline{A}C$
4. Simplify $A(B \oplus C) + \overline{A}(BC)$
5. How many solutions does this have? $AC(\overline{A} + B) + \overline{A}(B + C)$
6. Make a tree out of BIGPARSER
7. What is the internal path length of LORDNIKI
8. What is left in the stack from bottom to top following the operations?
   PUSH(N) PUSH(I) PUSH(K) PUSH(Y) POP(X) PUSH(I) PUSH(M) PUSH(E) PUSH(M) PUSH(E) POP(X) POP(X) POP(X) POP(X)
9. What is left in the queue from front to back following the operations?
   PUSH(M) POP(X) PUSH(E) PUSH(M) POP(X) PUSH(E) PUSH(S) POP(X) PUSH(L) PUSH(O) POP(X) PUSH(V) PUSH(E) PUSH(N) PUSH(I) PUSH(K) PUSH(I) POP(X)
10. Draw the FSA for 1*01U00*01U011*
11. Draw the FSA for (1*001*0)U(11*110U1)11*0
12. Does 1110110100100101010 match 11*1010*101001*0*1*(0*10)U(10*)1U01U0
13. Draw an FSA that can fit 110010101 1010010101010 and 10101010111 with no unions
14. Draw an FSA for a((bc*bbcc*b)*aa)*