Discrete Mathematics (SC612)
Tutorial 5
12^{th} November, 2021

1. (a) Convert 2148 in decimal system to:

   i. Base 2 (binary)
   solution:
   $2148 = 2048 + 64 + 32 + 4 = 2^{11} + 2^6 + 2^5 + 2^2$. Thus the binary representation of 2148 is

   $$100001100100$$

   ii. Base 3 (Ternary)
   solution:
   $2148 = 1458 + 486 + 162 + 27 + 9 + 6 = 2 \times 3^6 + 2 \times 3^5 + 2 \times 3^4 + 3^3 + 3^2 + 2 \times 3^1$. Thus the ternary representation of 2148 is

   $$2221120$$

   iii. Base 7
   solution:
   $2148 = 2058 + 49 + 35 + 6 = 6 \times 7^3 + 7^2 + 5 \times 7^1 + 6 \times 7^0$. Thus the Base 7 representation of decimal 2148 is

   $$6156$$

   iv. Base 8 (Octal)
   solution:
   $2148 = 2048 + 64 + 32 + 4 = 4 \times 8^3 + 8^2 + 4 \times 8^1 + 4 \times 8^0$. Thus the Octal representation of decimal 2148 is

   $$4144$$

(b) Which decimal system integer is represented by the base 7 number 632?

solution: Base 7: 632 is equivalent to:

$$6 \times 7^2 + 3 \times 7^1 + 2 \times 7^0 = 344 + 21 + 2 = 267$$

in decimal system.

(c) Which decimal system integer is represented by the base 8 (octal) number 577

solution:

Octal 577 is equivalent to:

$$5 \times 8^2 + 7 \times 8^1 + 7 \times 8^0 = 320 + 56 + 7 = 383$$

in decimal.

(d) Which decimal system number is represented by the base 3 (ternary) number 21022

solution:

Ternary 21022 is equivalent to:

$$2 \times 3^4 + 3^3 + 2 \times 3^1 + 2 \times 3^0 = 209$$

in decimal.

(e) Which decimal system number is represented by the base 2 (binary) number 10101110?

solution:

Binary 10101110 is equivalent to:

$$2^7 + 2^5 + 2^3 + 2^2 + 2^1 = 173$$

2. For each of the languages described below, construct a finite automaton where possible, and prove mathematically that none eists if that be the case.

(a) The language of strings over $\Sigma = \{a\}$ consisting of exactly those words of prime number length.

solution:

Not possible. Proof by contradiction. Assume that there is a machine for this language such that it has k states. Let t be the smallest prime $\geq k$. Take the string a^t. Since this is in the language, it must be accepted by the machine. The run of the machine is a state sequence of $t+1$ states on this word. This exceeds the number of distinct states of the machine. Thus there must be a repetition of states in the state sequence. Let the number of steps betweek successive repeats of an a is r. Thus the machine can increase or decrease the current run by multiples of r without affecting acceptance by the machine. This means all words of the type a^{t+i*r} and $t + ir$ is not a prime for all values of i. Such a word should not be accepted, but it is.

(b) The language of all words over $\Sigma = \{a, b, c\}$ where the last character is a.

solution:

Yes. It is a 2-state machine with one start state and the other state is accepting. The transitions are transition by a takes you to the non-start state, and all other transitions take the machine to the start state.

(c) The set of all words over $\{a, b, c\}$ which are palindromes (that is the word is the same when read from left-to-right or right-to-left).

solution:

No. Again proof by contradiction. Assume there is a machine for this language with k states. Take the string $a^k ba^k$ which is a palindrome. The initial run on the first a^k will see a repeated state (pigeon hole principle). If the string in the repeated portion is $a^t, t > 0$, then the word $a^k + tba^k$ is also accepted, but this word is not a palindrome.

(d) The set of all words over $\Sigma = \{a, b, c\}$ consisting of all those words where no two identical characters in the word are separated by a different character.

solution:

Yes. The automaton is too big to draw here. There will be states for:

- Seen no character
- Seen only a's

- Seen only b's
- Seen only c's
- Seen b after a
- Seen a after b ...

If at any point one sees a symbol seen in the past but not the latest symbol (indicated by the state the machine is in) go to a dead state and self-loop on all later characters. This is a rejecting state. In fact all other states are accepting. In effect we reject a word the moment an earlier occurence of a letter and the current occurrence has been interrupted by other characters occurring.

3. (a) Suppose L is a language over a finite alphabet $\Sigma = \{a, b\}$, such that $|L|$ is a finite number. Show that it has a finite automaton recognising it.

   solution:

   Since it has finitely many words, we can form what is called a prefix tree, including each word and convert this tree into an automaton by suitably folding, after the longest word in the language. Thus, any language with finitely many words is a formal language.

   (b) Suppose L is a language over $\Sigma = \{0, 1, 2, 3\}$ which is recognised by a finite automaton. Prove that the complement language $\Sigma^* \setminus L$ is also accepted by a finite automaton.

   solution:

   Just exchange the accepting and non-accepting states in the original machine to get a machine for the complement.

   (c) Suppose L_1 and L_2 are languages over the same finite alphabet that can both be recognised by finite automata (obviously different machines for the two). Show that the language $L_1 \cup L_2$ can also be recognised by a finite automaton.

   (d) Suppose L_1 and L_2 are languages over the same finite alphabet that can both be recognised by finite automata (obviously different machines for the two). Show that the language $L_1 \cap L_2$ can also be recognised by a finite automaton.

   solution:

For both the last two parts, the construction has many similarities. The set of states is the cartesian product of the states of the machines of the two individual languages. The alphabet is the same. The start state is the ordered pair for each machine. The transition is coordinate-wise as per the two individual machines. The accepting states is where the two machines differ. The union language machine accepts if **at least** one of the two coordinates is an accepting state of its machine. The intersection language machine accepts only if **both** the coordinates are accepting states in their respective machines.

HINT: Use the concept of cartesian product for parts (c) **and** (d)