PLEASE REMEMBER TO REGISTER FOR THE COURSE!

Instructions

- Classroom Problems C2.1–C2.2 will be discussed and solved onsite at the tutorial sessions in lecture week 2. No credit is given for these problems.
- Homework Problems H2.1–H2.3 you should solve on your own, and be available to present your solutions at one of the tutorial sessions in lecture week 3. In order to get course credit, you need to indicate your solved problems on the signup sheet circulated at the beginning of the session.
- Supplementary Problems S2.1–S2.3 provide further illustration and extension of the course material, but will usually not be covered at the tutorials. You are however invited to work on these problems too, and discuss them with the course staff. Sample solutions are provided on MyCourses.

Classroom Problems

C2.1 Design **complete** finite automata that recognise the following languages:

(i) $\{w \in \{a,b\}^* \mid w \text{ contains the symbol } a\}$;

(ii) $\{w \in \{a,b\}^* \mid w \text{ contains exactly two } a\text{'s } \}$;

(iii) $\{w \in \{0,1\}^* \mid w \text{ contains an even number (possibly zero) of 0's}\}$;

(iv) $\{w \in \{0,1\}^* \mid w \text{ begins with 01 and ends in a 0}\}$.

C2.2 Design a finite automaton (state machine) that models the behaviour of a simple TV set. The TV can be on or off, and when it is on, the channel selector of the TV has three positions (1/2/3), while the volume control has two (lo/hi). At the beginning the TV is off, but the automaton does not need to have any final states.

Homework Problems

H2.1 Design finite automata that recognise the following languages:

(i) $\{w \in \{a,b\}^* \mid w \text{ begins with the substring } aba\}$;

(ii) $\{w \in \{a,b\}^* \mid w \text{ contains } aba \text{ as a substring}\}$;

(iii) $\{w \in \{a,b\}^* \mid w \text{ has substring } aba \text{ at its end}\}$;

(iv) $\{w \in \{a,b\}^* \mid w$ contains both ab and ba as (possibly overlapping) substrings$\}$.

State (say) whether your automata are complete or not.

Your automata **don't have to be** complete (being complete is of course allowed, but not required).

H2.2 Design finite automata that recognise the following languages:

(i) $\{w \in \{0,1\}^* \mid$ the number of 1's in w is divisible by three (or possibly zero)$\}$;

(ii) $\{w \in \{0,1\}^* \mid$ the number of substrings 01 in w is odd$\}$;

(iii) $\{w \in \{0,1\}^* \mid$ the number of 0's in w is odd and the number of 1's in w is divisible by three (or possibly zero)$\}$.

State (say) whether your automata are complete or not.

Your automata **don't have to be** complete (being complete is of course allowed, but not required).

H2.3 Design a finite automaton that models the behaviour of a lift moving between two floors. The lift can be either up or down. Both floors have a simple 'call here' button for the lift, and inside the lift there are buttons for going 'up' and 'down'. In addition, the lift has a door that can be opened and closed; the lift only moves when the door is closed. The time required for the lift to travel between the two floors does not need to be taken into account, and any possible service requests occurring during this interval can be ignored. The automaton does not need to have any distinct "final states".

Supplementary Problems

S2.1 Formulate the model of a simple coffee machine presented at Lecture 2 (slide 2 of Sec. 2.1) precisely according to the mathematical definition of a finite automaton (slides 3–4 of Sec. 2.3). What is the formal language recognised by this automaton?

S2.2 Design finite automata that recognise the following languages:

(i) $\{a^m b^n \mid m = n \mod 3\}$;

(ii) $\{w \in \{a,b\}^* \mid w$ contains equally many a's and b's, modulo 3$\}$.

(The notation "$m = n \mod 3$" means that the numbers m and n yield the same remainder when divided by three.)

S2.3 Design a finite automaton that recognises sequences of nonnegative integers separated by plus and minus signs, where in addition the sequence may start with a minus sign (e.g. `11+20-9`, `-5+8`). Implement your automaton as a computer program that also calculates the numerical value of the input expression.