Discrete Event Systems

Exercise Session 1

Roland Schmid

nsg.ee.ethz.ch

ETH Zürich (D-ITET)

19 September 2024

1 Finite Automata

In this exercise you are asked to design your first finite automata. Try to minimize the number of states of your machines.

a) Consider the alphabet {0,1}. Implement an automaton which accepts the following strings:

At first there are zero or more '1's, followed by zero ore more '0's. This in turn is followed by an arbitrary (non-zero) number of '1's.

b) Design an automaton which decides whether a number is divisible by three. Assume that the digits of the number are inserted sequentially, that is, the number '135' is inserted as '1', '3' and finally '5'. How many states do you need? (Hint: Cross sum!)

Formal Definition of a Finite Automaton

A finite automaton (FA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$,where

- Q is a finite set called the states
- Σ is a finite set called the alphabet
- $\delta: Q \times \Sigma \rightarrow Q$ is the transformation function
- $q_0 \in Q$ is the start state
- $F \subseteq Q$ is the set of accept states (a.k.a. final states).

1 Finite Automata

In this exercise you are asked to design your first finite automata. Try to minimize the number of states of your machines.

a) Consider the alphabet {0,1}. Implement an automaton which accepts the following strings:

At first there are zero or more '1's, followed by zero ore more '0's. This in turn is followed by an arbitrary (non-zero) number of '1's.

b) Design an automaton which decides whether a number is divisible by three. Assume that the digits of the number are inserted sequentially, that is, the number '135' is inserted as '1', '3' and finally '5'. How many states do you need? (Hint: Cross sum!)

2 Vending Machine Revisited

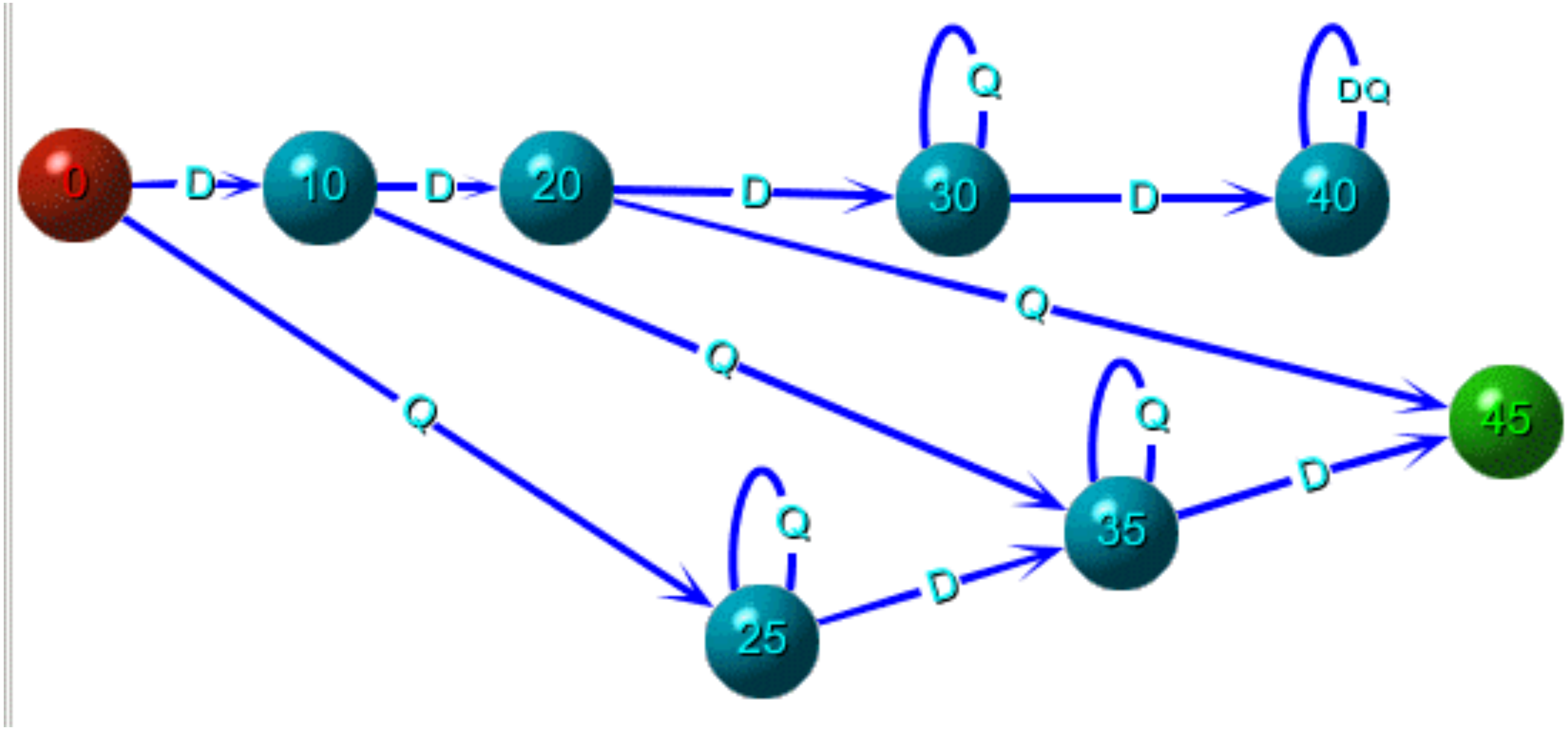
Consider the vending machine shown at the beginning of the course. Do you see any problems with this machine? How can the machine be made more user-friendly?

Vending Machine Java Code

```
Soda vend(){
   int total = 0, coin;
   while (total != 45){
        receive(coin);
        if ((coin==10 && total==40)
        ||(coin==25 && total>=25))
              reject(coin);
        else
              total += coin;
   }
   return new Soda();
}
```

Overkill?!?

Vending Machine “Logics”



3 “Mais im Bundeshuus”

It is Wednesday morning and the seven members of the Swiss Federal Council meet to decide about an important topic: In order to decrease the expenses of education, should only women be allowed to study at ETH?

a) Assume that the seven members vote one after another. Further, assume that there is no abstention of voting. Design an automaton which accepts the ballot if and only if the majority of the members voted in favor of the proposition.

b) Extend your automaton for the case of abstentions of voting. The automaton should accept if and only if more members voted in favor of the proposition than against it.

4 Build Finite Automata with the Construction Rules

Consider the alphabet {a,b}. For each of the following languages, implement an automaton that accepts it using the constructions seen in the lecture (*e.g.*, the cartesian product construction).

a) $\{w — w$ has exactly two `a`'s and at least two `b`'s$\}$

b) $\{w — w$ does not contain string `baba`$\}$

Cartesian Product Construction

- We want to construct a finite automaton M that recognizes any strings belonging to L_1 or L_2.

- Idea: Build M such that it simulates *both* M_1 and M_2 simultaneously and accept if either of the automatons accepts

Formal Definition

- Given two automata
  $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

- Define the unioner of M_1 and M_2 by:
  $M_\cup = (Q_1 \times Q_2, \Sigma, \delta_1 \times \delta_2, (q_1, q_2), F_\cup)$

  - where the accept state (q_1, q_2) is the combined start state of both automata

  - where $F_\cup$ is the set of ordered pairs in $Q_1 \times Q_2$ with at least one state an accept state. That is: $F_\cup = Q_1 \times F_2 \cup F_1 \times Q_2$

  - where the transition function δ is defined as
  $\delta\big((q_1, q_2), j\big) = \big(\delta_1(q_1, j), \delta_2(q_2, j)\big) = \delta_1 \times \delta_2$

Other constructions: Intersector

- Other constructions are possible, for example an intersector:
- Accept only when both ending states are accept states. So the only difference is in the set of accept states. Formally the intersector of M_1 and M_2 is given by
  $M_\cap = \left(Q_1 \times Q_2, \Sigma, \delta_1 \times \delta_2, \left(q_{0,1}, q_{0,2}\right), F_\cap\right)$, where $F_\cap = F_1 \times F_2$.

Complement

- How about the complement? The complement is only defined with respect to some universe.

- Given the alphabet Σ, the *default universe* is just the set of all possible strings Σ^*. Therefore, given a language L over Σ, i.e. $L \subseteq \Sigma^*$ the complement of L is $\Sigma^* - L$

- Note: Since we know how to compute set difference, and we know how to construct the automaton for Σ^* we can construct the automaton for $\overline{L}$.

- Question: Is there a simpler construction for $\overline{L}$?

- Answer: Just switch accept-states with non-accept states.

4 Build Finite Automata with the Construction Rules

Consider the alphabet {a,b}. For each of the following languages, implement an automaton that accepts it using the constructions seen in the lecture (*e.g.*, the cartesian product construction).

a) $\{w$ — w has exactly two `a`'s and at least two `b`'s$\}$

b) $\{w$ — w does not contain string `baba`$\}$

5 Exam question [2015]

Consider the alphabet $\Sigma = \{0, 1\}$ of binary strings representing numbers (with the most significant bit to the left). Draw a DFA which accepts all the strings that are divisible by 5. Strings are read from left to right. For example, the strings `0`, `101`, `1010` and `01010` would be accepted while `01`, `10` would not. Observe that the empty string, ε, should not be accepted either.

Hint: Consider relying on the modulo operator (mod) which returns the remainder of the division of one number by another. For instance, $1 \bmod 5 = 1$ and $2 \bmod 5 = 2$.

Reminder: A bit shift to the left doubles the value of a binary number.

6 Filter for an Input Stream

We would like to construct an automaton that recognizes substrings from an input stream. The input stream consists of symbols $\{a, b\}$ and the substrings that the automaton should detect are of the form bab^*. In other words, the input of the automaton is a series of a's and b's. The automaton should go into an accepting state whenever the most recently received symbols form a string of the form bab^*. For example, in the input stream $b\ \underline{a}\ \underline{b}\ \underline{b}\ \underline{b}\ \underline{a}\ a\ a\ a\ b\ \underline{a}\ \underline{b}\ \underline{a}\ a$, the automaton should be in an accepting state exactly after the reception of an underlined symbol. Construct a deterministic finite automaton that precisely fulfils the above specification.