Finite Automata

Regular Languages
We have a class of languages and ways of denoting them via regular expressions or graphs.
We need to answer the question: if I were to give you a string over some alphabet, is there a way to determine whether this string belongs to a particular language or not?
1. If we were given the RE for the language, we could attempt to show that the RE denotes the string; or that it cannot possibly do so.
2. If we were given the graph, we could try to thread the string through a path connecting the start node of the graph with the end node. If we find such a path, the graph “accepts” the string; if we can show no such path exists, then it “rejects” it.
We are going to “mimic” the graph...

Finite Automata

Deterministic Finite Automata
DFA: a simple machine that reads an input string one letter at a time and, after having read the input, decides to accept it or reject it.
DFA: (more specific) (Automata → singular Automaton)
1. A finite tape: used to store input data. Divided into a (finite) number of cells, with each cell holding a symbol from an alphabet Σ.
2. A tape head which scans the tape, reads the symbols, moves one cell to the right after each reading, and passes the information to
3. A finite control: consisting of a finite number of states (Q).
At the beginning of each move, the control is in one of the states. The state and the symbol under the tape head determine the next state (by a total transition function: δ : Q × Σ → Q).

A Finite Automaton

This doesn’t quite look like a graph, yet...

Finite Automata

Deterministic Finite Automata
The set of states Q contains a special state q₀, called the initial state, and a subset F ⊆ Q called the final states.
DFA: (formal def.): a quintuple
M = (Q, Σ, δ, q₀, F) = (States, Alphabet, Transition Function, Start State, Final States).
M starts in state q₀, with the read head over the leftmost cell of the input on the tape. It halts after it reads the symbol in the rightmost cell of the tape, and the tape head moves off the tape. It accepts the input string if it halts in one of the accepting states, it rejects the input otherwise.
Def.: the set of strings accepted by a DFA M is denoted by L(M). The language L(M) is accepted by the DFA M.
Finite Automata

Deterministic Finite Automata - Transition Diagrams

A transition diagram representation of a DFA $M$ is a labeled digraph $(V, E)$, satisfying

$$V \equiv Q$$
$$E \equiv \{q \rightarrow p \mid p = \delta(q, a)\},$$

where $q \rightarrow p$ denotes an edge $(q, p)$ with label $a$.

The initial state of the DFA is pointed to by an arrow without a starting vertex, and every final state is denoted by double circles.

We DO have a graph, after all...

From Quintuple to Graph (transition diagram)

Ex. 2.1. $M = (Q, \Sigma, \delta, q_0, F)$; $Q = \{q_0, q_1, q_2, q_3\}$; $\Sigma = \{0, 1\}$; $F = \{q_1, q_2\}$

The transition diagram of $M$ is shown in the figure.

Observations: for each string $x$ there exists a unique path starting from $q_0$ whose labels form $x$. The path is called the computation path of the DFA on $x$. A string $x$ is accepted by $M$ iff its computation path ends at a final state.

Languages from DFAs

Language (= set of strings) accepted by a DFA

We first extend $\delta$ from the domain $Q \times \Sigma$ to $Q \times \Sigma^*$ by induction as follows:

1. $\delta(q, \epsilon) = q$
2. For any $x \in \Sigma^*$ and $a \in \Sigma$, $\delta(q, xa) = \delta(\delta(q, x), a)$.

With this extension, for any string $x$, $\delta(q_0, x)$ is the last state in the computation path of $x$. This allows us another definition:

$$L(M) = \{x \in \Sigma^* \mid \delta(q_0, x) \in F\}$$

We have just shown that to each DFA there corresponds a (unique) language. The next question is: is this correspondence bi-unique? (to each language there also corresponds a unique DFA)

Languages from DFAs - Example

What is the language (= an RE denoting the language, or a verbal description of the language) accepted by the DFA

We observe that the string 0 terminates in an accepting state, the string 00 does so, as well as any string in $00(0+1)^*$. Any string starting with a 1 or a 01 will terminate in a non-accepting state.

$$L(M) = 0 + 00(0 + 1)^*$$ : RE description.
Finite Automata

The Other Direction: DFAs from Languages

Ex. 2.7. Find a DFA that accepts all binary strings beginning with 01 and no others (i.e. L(01(0+1)*)).

The labeled digraph can be constructed:

This will accept the strings of the language, reject the empty and 0 strings, but hang on others. We need to complete the transition function - once we do that, we have

Ex. 2.8. The set of all binary strings having a substring 00.

The corresponding regular expression is (0 + 1)*00(0 + 1)*.

The digraph construction will not be particularly useful, since it would involve four $\varepsilon$-transitions - we could collapse them via Thm. 1.25, BUT... Problem: why?

Ex. 2.8. The set of all binary strings having a substring 00.

The transitions in a DFA must occur on all actual elements of $\Sigma$ and only elements of $\Sigma$. Notice that we would like to find out when the first occurrence of 00 has taken place - at that point, we are done. We start with a graph that will accept some strings in the language (the ones that start with 00).

What else can happen? A string can start with a 1, or with a 0 immediately followed by a 1. In either case, we are still waiting for the 00. How can we express this?
Both of the previous examples

- Start with a transition graph that captures many strings that belong to the language.
- Are completed by filling in the rest of the transitions.

**General construction:**

1. Let \( x_1x_2...x_n \) be the string we want to check for the presence of a substring \( y_1...y_k \).
2. Construct a DFA with states \( q_0, q_1, ..., q_k \), and transition function \( \delta \) such that \( \delta(q_0, y_1) = q_1, ..., \delta(q_{k-1}, y_k) = q_k \). \( q_0 \) is the initial state, \( q_k \) is the accepting state.

3. We still need to complete the transitions for \( \delta(q_j, y) \), \( y \neq y_j \).

These transitions will depend on the substring and on the language.

**Example:** the set of all binary strings having the substring 00101.

**Step 1:** a "recognizer" for the substring \( y_1...y_k = 00101 \).

**Step 2:** fix the end - what happens after you have seen 00101?

**Step 3:** what happens if you see the "wrong character" for the state? You are inside the machine that accepts 00101 and see a character that is not a proper continuation: BACKTRACK to the appropriate state (this part requires some thought).
Finite Automata

DFAs from Languages

Transition Function (continued):
We define \( \delta(q_i, a) = q_j \), where \( j \equiv 2i + a \pmod{5} \).
State \( q_0 \) is the unique final state.
Since the binary expansion of a positive binary integer always starts with a 1, we need to add an initial state and a failure state.

DFAs from Languages

From the transition function \( \delta(q_i, a) = q_j \), where \( j \equiv 2i + a \pmod{5} \), we construct the graph DFA:

DFAs and Languages - Closure Properties

We look back at some results about regular expressions:
1. Since the disjunction of two regular expressions is a regular expression, we can conclude that the union of two regular languages is a regular language.
2. Since the concatenation of two regular expressions is a regular expression, we can conclude that the concatenation of two regular languages is a regular language.
3. Same for Kleene closure.
4. What about intersection, difference and complementation? There seems to be no immediate way of representing these operations as operations on regular expressions.... How about for DFAs???

Theorem 2.16. The class of languages accepted by a DFA is closed under union, intersection, subtraction and complementation.

Proof. union. Let \( L(M_1) \) and \( L(M_2) \) be the two languages with their corresponding automata. Determining whether a string \( x \in L(M_1) \cup L(M_2) \) could be done by "running" the string through both \( M_1 \) and \( M_2 \) simultaneously. We thus need a way to construct this "simultaneous" \((M_1, M_2)\) DFA. Define:
1. \( M = M_1 \times M_2 \); \( \{ (q_{i1}, q_{i2}) \mid q_{i1} \in M_1, q_{i2} \in M_2 \} \); the states.
2. \( \delta( (p, q), a ) = (\delta_{1}(p, a), \delta_{2}(q, a)) \) for \( p \in Q_1, q \in Q_2, a \in \Sigma \).
3. \( F = F_1 \times F_2 \cup F_1 \times Q_2 \).

If the alphabets were different, extend the original DFAs with transitions to "dead" states, so as to have a single alphabet.
Finite Automata

DFA and Languages - Closure Properties

It should be now clear that \( M \) will accept exactly \( L(M_1) \cup L(M_2) \).

**intersection.** The only change we need to make to the previous machine is a redefinition of the final states: \( F = F_1 \times F_2 \).

**subtraction.** Redefine: \( F = F_1 \times (Q_2 - F_2) \), to recognize \( L(M_1) - L(M_2) \).

**complementation.** \( \overline{A} = \Sigma^* - A \). If \( M = (Q, \Sigma, \delta, q_0, F) \), \( x \in L(M) \iff \delta(q_0, x) \in F \). So \( x \notin L(M) \iff \delta(q_0, x) \notin F \iff \delta(q_0, x) \in Q - F \). Redefine the set of accepting states to be \( Q - F \).