Chapter I Automata Theory, an Algorithmic Approach

1. Automata as Data Structures

- Data structures allow us to represent sets of objects in a computer.
- Different data structures support different sets of operations (dictionary, stack, queue, priority queue, . . .):

Op. set	Operations	Data structures
Dictionary	insert, lookup, remove	Hash tables, arrays, search trees
Stack	push, pop	Linked list, array
Priority queue	insert_with_priority, extract_highest_priority	Heap, binomial heap, Fibonacci heap
Union-find	set union, find set	Linked lists, disjoint forests

Automata as Data Structures

In this course we look at automata as a data structure supporting

- the boolean operations of set theory (union, intersection, complement with respect to a given universe set)
- property checks (emptiness, universality, inclusion, equality)
- operations on relations (projections, joins, pre, post)

1.1 Algorithmic Operations on Sets and Relations

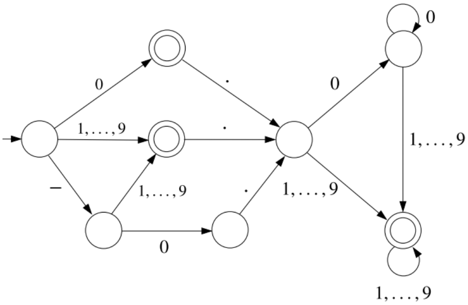
Member(x, X)	:	returns **true** if $x \in X$, **false** otherwise
Complement(X)	:	returns $U \setminus X$
Intersection(X, Y)	:	returns $X \cap Y$
Union(X, Y)	:	returns $X \cup Y$
Empty(X)	:	returns **true** if $X = \emptyset$, **false** otherwise
Universal(X)	:	returns **true** if $X = U$, **false** otherwise
Included(X, Y)	:	returns **true** if $X \subseteq Y$, **false** otherwise
Equal(X, Y)	:	returns **true** if $X = Y$, **false** otherwise
Projection$_1(R)$	:	returns the set $\pi_1(R) = \{x;\ (\exists x)[(x, y) \in R]\}$
Projection$_2(R)$	:	returns the set $\pi_2(R) = \{y;\ (\exists y)[(x, y) \in R]\}$
Join(R, S)	:	returns $R \circ S = \{(x, z);\ (\exists y)[(x, y) \in R \wedge (y, z) \in S]\}$
Post(X, R)	:	returns post$_R(X) = \{y \in U;\ (\exists x \in X)[(x, y) \in R]\}$
Pre(X, R)	:	returns pre$_R(X) = \{y \in U;\ (\exists x \in X)[(y, x) \in R]\}$

Basic Idea

- Elements of the universe can be encoded as words (strings over some alphabet)
- Sets can be encoded as languages (sets of words)
- Automata recognize languages

Example 5

A finite automaton for the strings encoding decimal numbers:



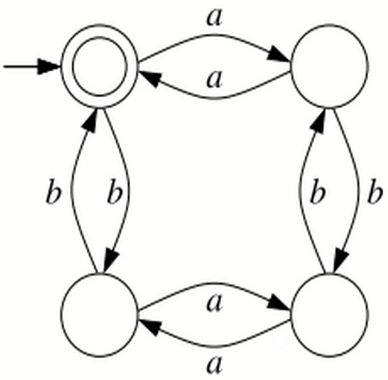
This is a first attempt! What can be corrected/improved?

1.2 Classes of Finite Automata

In the following, we show the definitions of

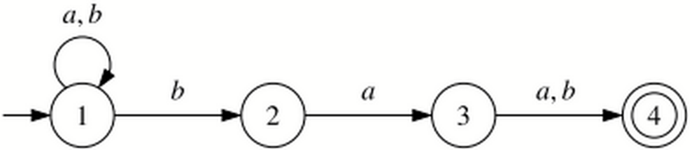
- deterministic finite automata (DFA)
- nondeterministic finite automata (NFA)
- nondeterministic finite automata with ϵ-transitions (NFA-ϵ)
- nondeterministic finite automata with regular-expression-transitions (NFA-reg)

Deterministic finite automata (DFA)



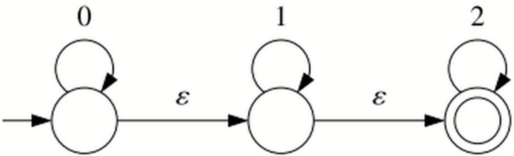
- Q *is a set of* states,
- Σ *is an alphabet,*
- $\delta\colon Q \times \Sigma \rightarrow Q$ *is a* transition function,
- $q_0 \in Q$ *is the* initial state, *and*
- $F \subseteq Q$ *is the set of* final states.

Nondeterministic finite automata (NFA)



- $\delta: Q \times \Sigma \rightarrow \mathcal{P}(Q)$ *is a* transition relation.

Nondeterministic finite automata with epsilon-transitions (NFA-e)



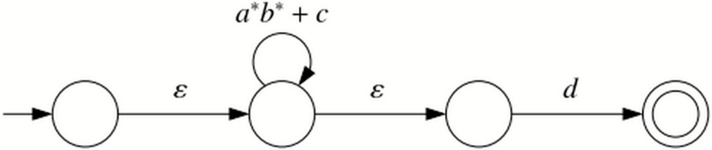
- $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$ *is a* transition relation.

Regular expressions

$$r ::= \emptyset \mid \varepsilon \mid a \mid r_1 r_2 \mid r_1 + r_2 \mid r^* \qquad \textit{where } a \in \Sigma$$

- $L(\emptyset) = \emptyset$,
- $L(\varepsilon) = \{\varepsilon\}$,
- $L(a) = \{a\}$,
- $L(r_1 r_2) = L(r_1) \cdot L(r_2)$, $\qquad L_1 \cdot L_2 = \{w_1 w_2 \in \Sigma^* \mid w_1 \in L_1, w_2 \in L_2\}$
- $L(r_1 + r_2) = L(r_1) \cup L(r_2)$,
- $L(r^*) = L(r)^*$. $\qquad L^* = \bigcup_{i \geq 0} L^i$, where $L_0 = \{\varepsilon\}$ and $L_{i+1} = L^i \cdot L$

Nondeterministic finite automata with regular-expression transitions (NFA-reg)



- $\delta\colon Q \times \mathcal{RE}(\Sigma) \to \mathcal{P}(Q)$ *is a relation such that* $\delta(q, r) = \emptyset$ *for all but a finite number of pairs* $(q, r) \in Q \times \mathcal{RE}(\Sigma)$.