Introduction to Computer Science

Géza Horváth

Third Lecture

Finite automata – definition

Definition (Finite automata)

Let $A = (Q, T, q_0, \delta, F)$. It is a finite automaton (recognizer), where Q is the finite set of (inner) states, T is the input (or tape) alphabet, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final (or accepting) states and δ is the transition function as follows.

- $\delta : Q \times T \to Q$ *(for deterministic finite automata);*
- $\delta : Q \times T \to 2^Q$ *(for nondeterministic finite automata).*

Finite automata – description

One can imagine a finite automaton as a machine equipped with an input tape. The machine works on a discrete time scale. At every point of time the machine is in one of its states, then it reads the next letter on the tape (the letter under the reading head), and then, according to the transition function (depending on the actual state and the letter being read,) it goes to the next state. It may happen in some variations that there is no transition defined for the actual state and letter, then the machine gets stuck and cannot continue its run. This case the automaton is called **partially defined** finite automaton. Otherwise the automaton is called **completely defined** finite automaton.

Finite automata – example

Example

Let the finite automaton A be

$A = (\{q_0, q_1, q_2, q_3\}, \{a, b, c\}, q_0, \delta, \{q_2, q_3\})$, *where*
$\delta(q_0, a) = \{q_1\}$,
$\delta(q_0, b) = \{q_0\}$,
$\delta(q_0, c) = \{q_0\}$,
$\delta(q_1, a) = \{q_1\}$,
$\delta(q_1, b) = \{q_0\}$,
$\delta(q_1, c) = \{q_2\}$,
$\delta(q_2, a) = \{q_2, q_3\}$,
$\delta(q_3, b) = \{q_3\}$,
$\delta(q_3, c) = \{q_1, q_2, q_3\}$.

See the explanation on the whiteboard.

Finite automata representation – Cayley table

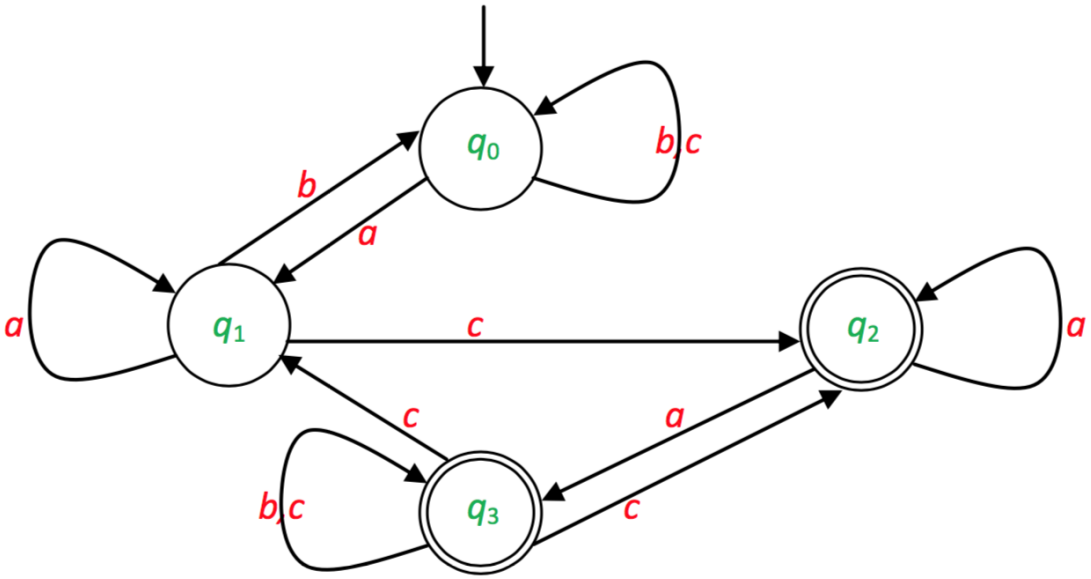
Example (Finite automata – Cayley table)

Let the finite automaton A be

T Q	$\rightarrow q_0$	q_1	$\overline{q_2}$	$\overline{q_3}$
a	q_1	q_1	q_2, q_3	$-$
b	q_0	q_0	$-$	q_3
c	q_0	q_2	$-$	q_1, q_2, q_3

When an automaton is given by a Cayley table, then the 0th line and the 0th column of the table are reserved for the states and for the alphabet, respectively (and it is marked in the 0th element of the 0th row).
The initial state should be the first among the states (it is advisable to mark it by a $\rightarrow$ sign also). The final states should also be marked, they should be circled.
The transition function is written into the table: the elements of the set $\delta(q, a)$ are written (if any) in the field of the column and row marked by the state q and by the letter a.

Finite automata representation – graph



Automata can also be defined in a graphical way: let the vertices (nodes, that are drawn as circles in this case) of a graph represent the states of the automaton (we may write the names of the states into the circles). The initial state is marked by an arrow going to it not from a node. The accepting states are marked by double circles. The labeled arcs (edges) of the graph represent the transitions of the automaton. If $p \in \delta(q, a)$ for some $p, q \in Q$, $a \in T$, then there is an edge from the circle representing state q to the circle representing state p and this edge is labeled by a.

Language accepted by finite automaton

Definition (Language accepted by finite automaton)

Let $A = (Q, T, q_0, \delta, F)$ be an automaton and $w \in T^*$ be an input word. We say that w is accepted by A if there is a run of the automaton, i.e., there is an alternating sequence $q_0 t_1 q_1 \dots q_{k-1} t_k q_k$ of states and transitions, that starts with the initial state q_0, ($q_i \in Q$ for every i, they are not necessarily distinct, e.g., $q_i = q_j$ is allowed even if $i \neq j$) and for every of its transition t_i of the sequence

- $t_i : q_i \in \delta(q_{i-1}, a_i)$ *in nondeterministic cases,*
- $t_i : q_i = \delta(q_{i-1}, a_i)$ *in deterministic cases,*

where $a_1 \dots a_k = w$, and $q_k \in F$. This run is called an accepting run.

All words that A accepts form $L(A)$, the language accepted (or recognized) by the automaton A.

Finite automata - determinization

Theorem (Determinization of finite automata)

For every finite automaton there is an equivalent (completely defined) deterministic finite automaton.

Proof. The proof is constructive. Let $A = (Q, T, q_0, \delta, F)$ be a nondeterministic finite automaton.
Let us construct the automaton $A' = (Q', T, q_0', \delta', F')$, where $Q' \subseteq 2^Q$, contains each possible subset of 2^Q which can be reached from q_0 and it also contains a new element q_e.
$q_0' = \{q_0\}$ and $F' \subset Q'$ includes every element $q' \in Q'$ such that $q' \cap F \neq \emptyset$.
The transition function δ' is defined as follows:

- $\delta'(q_e, a) = q_e$ for any $a \in T$, and
- if $\bigcup_{q \in q'} \delta(q, a) = \emptyset$ then $\delta'(q', a) = q_e$, else
- $\delta'(q', a) = \bigcup_{q \in q'} \delta(q, a)$, for any $a \in T$ and $q' \in Q'$.

Finite automata - determinization

One can observe that A' is a completely defined deterministic automaton. Also, every run of A has an equivalent run of A', in fact, A' simulates every possible run of A on the input at the same time. Conversely, if A' has an accepting run, then A also has at least one accepting run for the same input. Therefore, A and A' accept the same language, consequently they are equivalent. QED.

See the example on the whiteboard.

Equivalence of finite automata and regular grammar

Theorem

Every language generated by a regular grammar is accepted by a finite automaton.

Proof. The proof is constructive. Let $G = (N, T, S, P)$ be a regular grammar in regular normal form. Then, let the finite automaton

$$A = (Q, T, q_0, \delta, F)$$

be defined as follows:
let $Q = N \cup \{F'\}$ (where $F' \notin N$),
$q_0 = S$.
Let the transition function δ be defined by the elements of P: let $B \in \delta(A, a)$ if $A \to aB \in P$; and let $F' \in \delta(A, a)$ if $A \to a \in P$. Further, let the set of accepting states be $\{F'\}$ if $S \to \lambda$ is not in P and let $F = \{F', S\}$ if $S \to \lambda \in P$.

Equivalence of finite automata and regular grammar

One can see that the successful derivations in the grammar and the accepting runs of the automaton have a one-to-one correspondence.
QED.

See the example on the whiteboard.

Equivalence of finite automata and regular grammar

Theorem

Every language accepted by a finite automaton can be generated by a regular grammar.

See the proof + example on the whiteboard.

Corollary

A language is regular, if and only if it is accepted by some finite automaton.

References

This work was supported by the construction EFOP-3.4.3-16-2016-00021. The project was supported by the European Union, co-financed by the European Social Fund.