CSC236 winter 2020, week 10: Finite automata

Recommended reading: Chapter 7 Vassos course notes

Colin Morris
colin@cs.toronto.edu
http://www.cs.toronto.edu/~colin/236/W20/

March 16, 2020

Important reading

http://www.cs.toronto.edu/~colin/236/W20/last3weeks/

Supplemental reading/viewing

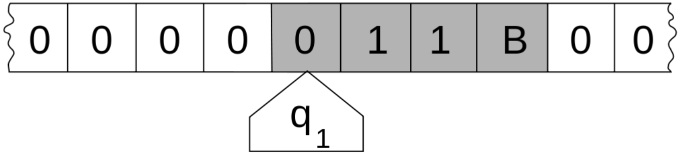
- Vassos course notes, Chapter 7
- David Liu course notes (final chapter, starts pg. 63)
- David Liu also has some excellent YouTube videos on 236 topics

- Wed 12-2

A "purely mechanical process"

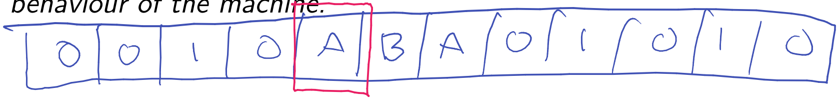
It was stated above that 'a function is effectively calculable if its values can be found by some purely mechanical process'. We may take this statement literally, understanding by a purely mechanical process one which could be carried out by a machine. It is possible to give a mathematical description, in a certain normal form, of the structures of these machines.

(Alan Turing, 1939)



Turing machine

We may compare a man in the process of computing a real number to a machine which is only capable of a finite number of conditions $\underline{q_1, q_2, \dots q_R}$ *which will be called "m-configurations". The machine is supplied with a "tape" (the analogue of paper) running through it, and divided into sections (called "squares") each capable of bearing a "symbol". At any moment there is just one square, say the r-th, bearing the symbol* $\mathcal{G}(r)$ *which is "in the machine". We may call this square the "scanned square ". The symbol on the scanned square may be called the "scanned symbol". The "scanned symbol" is the only one of which the machine is, so to speak, "directly aware". However, by altering its m-configuration the machine can effectively remember some of the symbols which it has "seen" (scanned) previously. The possible behaviour of the machine at any moment is determined by the m-configuration* q_n *and the scanned symbol* $\mathcal{G}(r)$*. This pair* q_n, $\mathcal{G}(r)$ *will be called the " configuration": thus the configuration determines the possible behaviour of the machine.*



> *In some of the configurations in which the scanned square is blank (i.e. bears no symbol) the machine writes down a new symbol on the scanned square: in other configurations it erases the scanned symbol. The machine may also change the square which is being scanned, but only by shifting it one place to right or left. In addition to any of these operations the m-configuration may be changed. Some of the symbols written down will form the sequence of figures which is the decimal of the real number which is being computed. The others are just rough notes to "assist the memory".*

Finite state automata — like Turing machines, but without the "rough notes"

Finite state automata

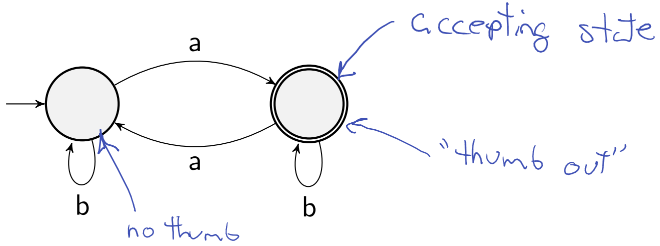
AKA finite state machines, AKA deterministic finite state automata, AKA DFSAs, DFAs...

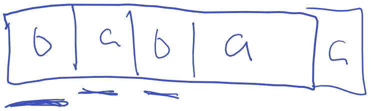
Example: A machine accepting ODDA $= \{s \in \{a, b\}^* \mid$ s has an odd number of a's$\}$

Finite state automata

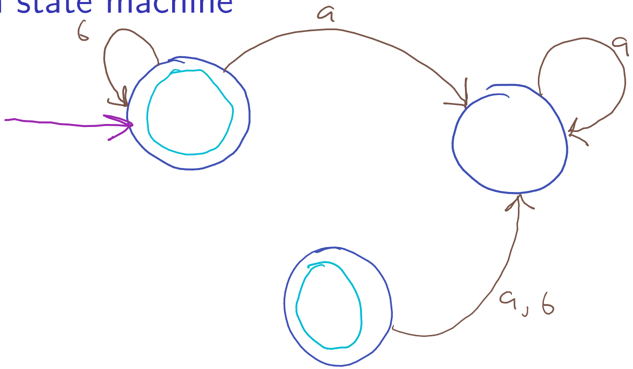
AKA finite state machines, AKA deterministic finite state automata, AKA DFSAs, DFAs...

Example: A machine accepting ODDA $= \{s \in \{a, b\}^* \mid$ s has an odd number of a's$\}$





Anatomy of a state machine



states

transitions

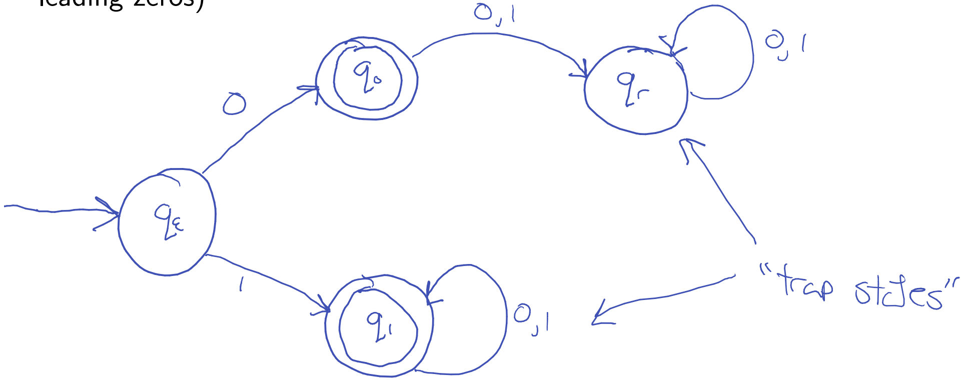
starting state

accepting state

Another example: BIN

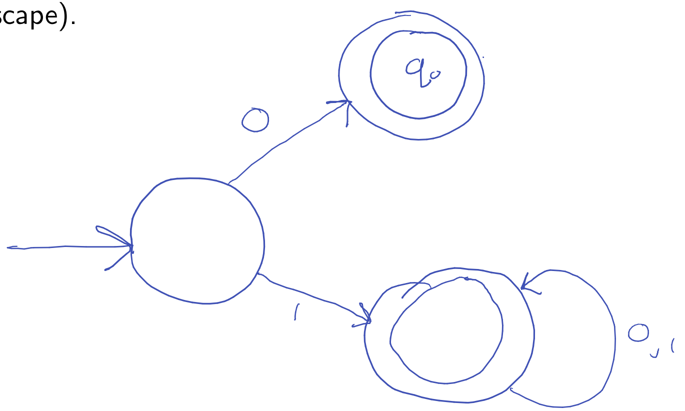
$0 + (1(1+0)^*)$

Recall from last lecture, BIN, the language of binary numbers (with no redundant leading zeros)



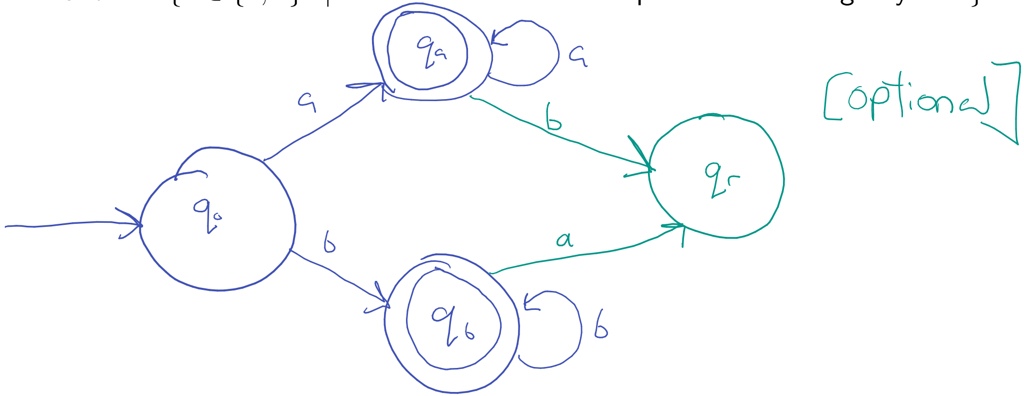
Convention: implicit dead states

When drawing state diagrams, if we don't draw a transition for symbol *a* from state *q*, it's assumed to go to a **dead state** (a non-accepting state from which there is no escape).



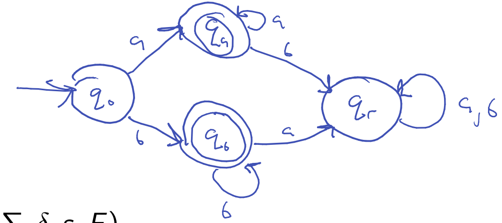
Your turn: UNIFORM

$\text{UNIFORM} = \{s \in \{a, b\}^* \mid s$ consists of non-zero repetitions of a single symbol$\}$.



Formal definition

$M =$



A DFSA M is a quintuple, $M = (Q, \Sigma, \delta, s, F)$

Q finite set of states $\longrightarrow \{q_0, q_a, q_b, q_r\}$

Σ finite alphabet $\Sigma = \{a, b\}$

$\delta : Q \times \Sigma \to Q$ the transition function

$s \in Q$ start state $s = q_0$

$F \subseteq Q$ set of accepting states $F = \{q_a, q_b\}$

The transition function, δ

$\delta = \{ ((\underline{q_0, a}), \underline{q_a}), \ldots$

$\delta : Q \times \Sigma \to Q$

$\delta(q_n, a)$ answers the question "where do we go if we're in state q_n and we see the symbol a?"

$\delta(q_0, a) = q_a$

	a	b
q_0	q_a	q_b
q_a	q_a	q_r
q_b	q_r	q_b
q_r	q_r	q_r

for UNIFORM DFA

The *extended* transition function, δ^*

$\delta^* : \underline{Q} \times \underline{\Sigma^*} \rightarrow Q$.

$\delta^*(q_n, x)$ answers the question "where do we end up if we start from state q_n and process all the symbols in string x?"

A string $x \in \Sigma^*$ is **accepted** by FSA M iff $\delta^*(s, x) \in F$.

$\mathcal{L}(\cancel{D})$ $L(M)$ is the language containing the strings accepted by FSA M.

$\mathcal{L}(0/^*) = \ldots$

$\delta^*(aaa) = q_a$ ✓

$\delta^*(\varepsilon) = q_0$ ✗

$\delta^*(aaabaaa) = q_r$ ✗

Proving correctness of automata

cf. 7.3.3 in Vassos notes

WTS: $\mathcal{L}(D) = \text{UNIFORM}$ $= \{a, b, aa, bb, aaa \ldots\}$

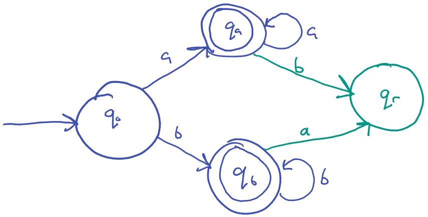
In other words, WTS the following predicate holds $\forall x \in \{a, b\}^*$:

$P(x)$: D accepts $x \iff x$ is of the form a^k or b^k for some $k > 0$.

Basis: ε, $P(\varepsilon)$ ✓

Assume $P(x)$ for $x \in \{a, b\}^*$

WTS: $P(xa) \wedge P(xb)$



Strengthening our predicate

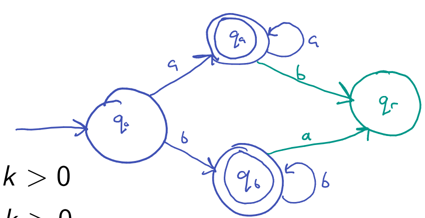
$\delta^*(q, \varepsilon) = q$

Basis $\delta^*(s, \varepsilon) = q_0$, $s = q_0$

$P(\varepsilon)$ holds

$$P(x) : \delta^*(s, x) = \begin{cases} q_0 & \text{if } x = \varepsilon \\ q_a & \text{if } x = a^k, \text{ for some } k > 0 \\ q_b & \text{if } x = b^k, \text{ for some } k > 0 \\ q_r & \text{otherwise (i.e. x contains a's and b's)} \end{cases}$$

state invariants



IS

Assume $P(x)$ for some x

WTS: $P(xa) \wedge P(xb)$

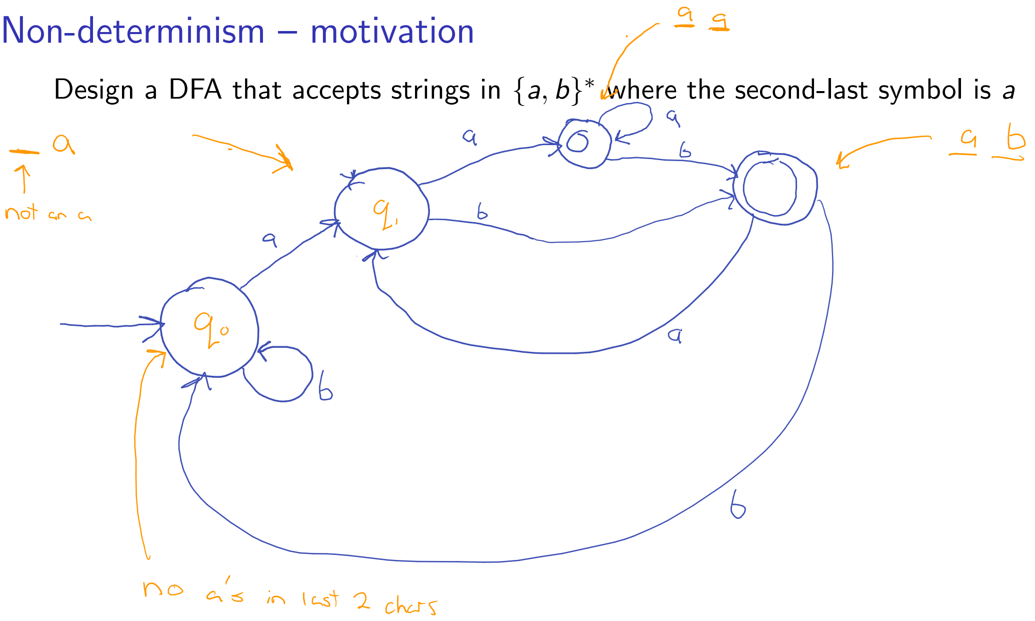
$\delta^*(s, xa) = \delta(\delta^*(s, x), a)$

By IH, we know

$$\delta^*(s, xa) = \begin{cases} \delta(q_0, a) = q_a & \text{if } x = \varepsilon \\ \delta(q_a, a) = q_a & \text{if } x = a^k, k > 0 \\ \delta(q_b, a) = q_r & \text{if } x = b^k \\ \delta(q_r, a) = q_r & \text{if } x \text{ contains a's and b's} \end{cases}$$

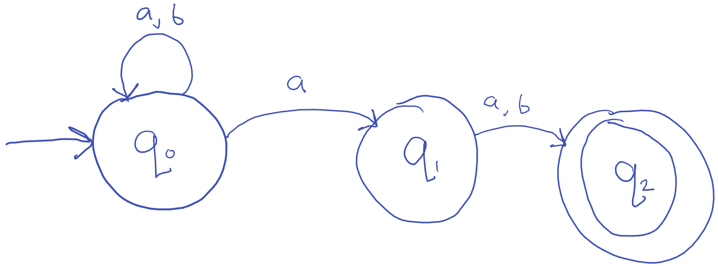
Non-determinism – motivation

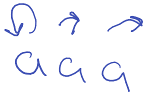
Design a DFA that accepts strings in $\{a, b\}^*$ where the second-last symbol is a



The non-deterministic way (penultimate a's)

"NFA"





RE: $(a+b)^* a (a+b)$

From $\underline{\text{DFA}}$ to $\underline{\text{NFA}}$

$M = (Q, \Sigma, \delta, s, F)$

Two new features:

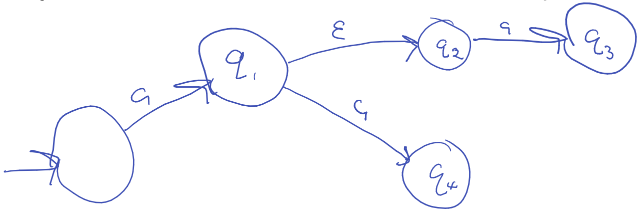
1. A state q can have **multiple** transitions when it sees symbol a
   - Instead of mapping to a specific state, δ (and δ^*) map to *sets of states*
2. ε **transitions** — we can have arrows between states labelled with the empty string, ε. These can happen 'spontaneously'

One change to formal definition:

- (previously) $\delta : Q \times \Sigma \to Q$
- (NFA) $\underline{\delta} : Q \times \underline{(\Sigma \cup \{\varepsilon\})} \to \underline{\mathcal{P}(Q)}$
  - Where $\mathcal{P}(Q)$ denotes the **powerset** of Q — the set of all subsets. Also written 2^Q.

A use for ε transitions

Let M_1 and M_2 be arbitrary automata. Construct an NFA that accepts $\mathcal{L}(M_1) \cup \mathcal{L}(M_2)$.

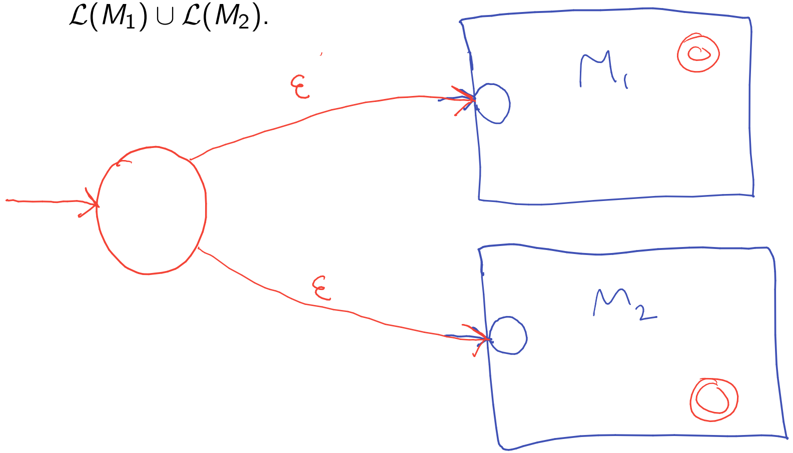


$x = aa$

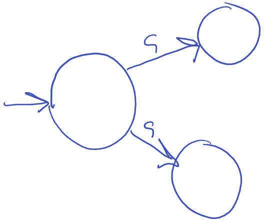
$\delta^*(s, aa) = \{q_4, q_3\}$

A use for ε transitions

Let M_1 and M_2 be arbitrary automata. Construct an NFA that accepts $\mathcal{L}(M_1) \cup \mathcal{L}(M_2)$.



Life in a non-deterministic world



An NFA accepts a string x if $\boxed{F \cap \delta^*(s, x) \neq \emptyset.}$
Take your choice of intuition:

- The NFA tries all possible paths at once
- The NFA 'magically' knows the right path to take (if one exists)

(Don't confuse non-determinism with stochasticity. The machine isn't rolling dice.)