Acceptance by DFA

Defining DFAs

- For each description let's draw a DFA that recognizes the language it describes

- { aa,ab,ac}
- $\{\Lambda$, a, abb, abbbb, … ,ab^{2n}, … }
- $\{a^m bc^n \mid m,n \in N\}$

Formal definition

- Recall a **D**FA is a quintuple $\mathbf{A} = (\mathbf{Q}, \Sigma, \mathbf{s}, \mathbf{F}, \mathrm{T})$, where
  - **Q** is a set of *states*
  - Σ is the alphabet of *input symbols*
  - **s** is an element of **Q** --- the *initial state*
  - **F** is a subset of **Q** ---the set of *final states*
  - T: $\mathbf{Q} \times \Sigma \longrightarrow \mathbf{Q}$ is the *transition function*

Example

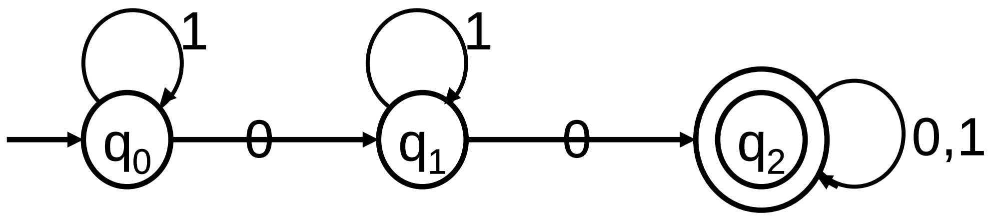
- In the example below,

Q$=\{q_0, q_1, q_2\}$,
$\Sigma=\{0,1\}$,
s$=q_0$,
F$=\{q_2\}$,

- anT

T is given by 6 equalities

$T(q_0,0)=q_1$,
$T(q_0,1)=q_0$,
$T(q_2,1)=q_2$
…



Transition Table

(Hein 11.2.6)

- All the information presenting a TFA can be given by a single thing -- its *transition table*:

	0	**1**
$\rightarrow$ $\mathbf{Q_0}$	Q_1	Q_0
$\mathbf{Q_1}$	Q_2	Q_1
$*\mathbf{Q_2}$	Q_2	Q_2

- The initial and final states are denoted by $\rightarrow$ and * respectively.

Extension of T to Strings

- Given a state **q** anT a string **w**, there is a unique path labeled **w** that starts at **q** (why?). The endpoint of that path is denoted $\underline{T}$**(q,w)**

- Formally, the function $\underline{T} : Q \times \Sigma^* \to Q$
- is defined recursively:

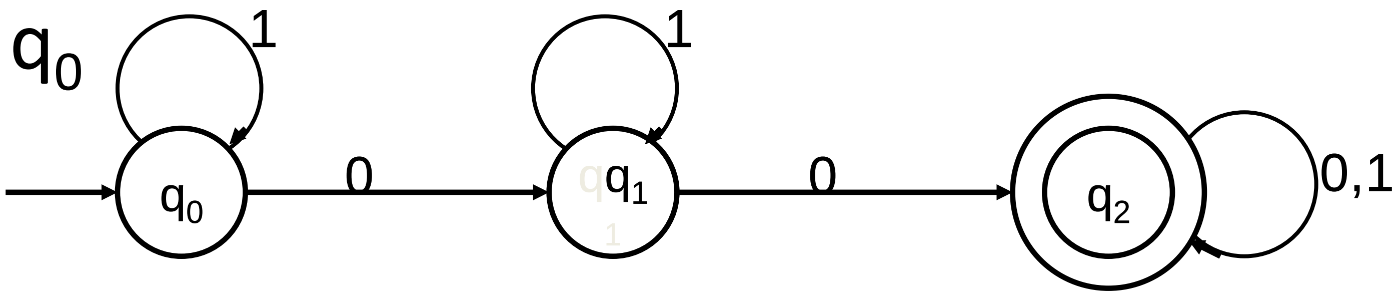
  – $\underline{\mathbf{T}}$**(q,ε)=q**
  – $\underline{\mathbf{T}}$**(q,ua)= T(**$\underline{\mathbf{T}}$**(q,u),a)**

- Note that $\underline{\mathbf{T}}$(q,a)= **T**(q,a) for every $a \in \Sigma$;
- so $\underline{\mathbf{T}}$ Toes extend **T**.

Example trace

- Diagrams (when available) make it very easy to compute $\underline{\mathtt{T}}(\mathtt{q},\mathtt{w})$ --- just trace the path labeled $\mathtt{w}$ starting at $\mathtt{q}$.

- E.g. trace 101 on the Diagram below starting at q_0



- Implementation and precise arguments need the formal definition.

$$
\begin{aligned}
\underline{\mathtt{I}}(q_0,101) &= T(\underline{\mathtt{I}}(q_0,10),1) \\
&= T(T(\underline{\mathtt{I}}(q_0,1),0),1) \\
&= T(T(T(q_0,1),0),1) \\
&= T(T(q_0,0),1) \\
&= T(q_1,1) \\
&= q_1
\end{aligned}
$$

	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_2	q_1
$*q_2$	q_2	q_2

Language of accepted strings

A DFA $=(\mathbf{Q},\Sigma,\mathbf{s},\mathbf{F},\mathrm{T})$, *accepts* a string $\mathbf{w}$ iff $\underline{\mathrm{T}}(\mathbf{s},\mathtt{w}) \in \mathbf{F}$

The language of the automaton A is

$\mathtt{L(A)=\{w \mid A}$ accepts $\mathtt{w\}}$.

More formally

$\mathtt{L(A)=\{w \mid \underline{T}(Start(A),w) \in Final(A)\}}$

Example:

Find a DFA whose language is the set of all strings over $\{a,b,c\}$ that contain aaa as a substring.

DFA's as Haskell Programs

```
data DFA q s = DFA { states :: [q],
                     symbols :: [s],
                     delta :: q -> s -> q,
                     start :: q,
                     final :: [q]}
```

Haskell is a functional language that makes it easy to describe formal (or mathematical) objects.

Transition function

```
trans :: (q -> s -> q) -> q -> [s] -> q
trans T q [] = q
trans T q (s:ss) = trans T (T q s) ss

accept :: (Eq q) => TFA q s -> [s] -> Bool
accept
  m@(TFA{Telta = T,start = q0,final = f}) w
  = elem (trans T q0 w) f
```

An Example

```
ma = DFA { states = [0,1,2],
           symbols = [0,1],
           delta = \p a ->
                    (2*p+a) `mod` 3,
           start = 0,
           final = [2]
         }
```

Another definition of acceptance

A DFA $A = (\mathbf{Q}, \Sigma, \mathbf{s}, \mathbf{F}, \mathrm{T})$, *accepts* a string $\mathbf{x_1 x_2 .. x_n}$ (an element of Σ^*) iff

- There exists a sequence of states $\mathbf{q_1 q_2 .. q_n q_{n+1}}$ such that
  1. $\mathbf{q_1 = s}$
  2. $\mathbf{q_{i+1}} = \mathrm{T}(\mathbf{q_i}, \mathbf{x_i})$
  3. $\mathbf{Q_{n+1}}$ is an element of $\mathbf{F}$

Note, one more state than characters in the input string

How does this relate to our previous definition?

$L(A) = \{w \mid \underline{\mathrm{T}}(\mathbf{s}, w) \in \mathbf{F}\}$