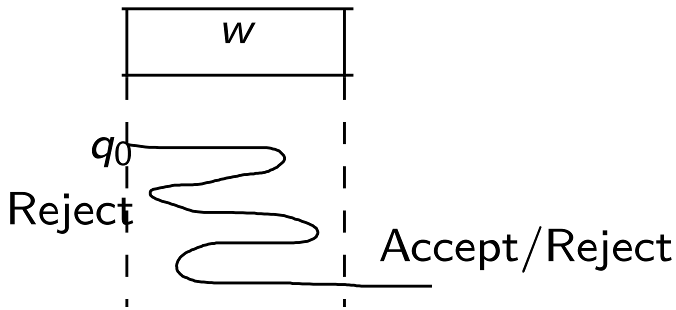
Further Generalisation

- Finite automaton makes following actions:
  - read a symbol
  - changes its state
  - moves its reading head to the right
- The head is not allowed to move to the left.

- What happens, if we allow the head to move left and right?
- **The automaton does not write anything on the tape!**



Two way finite automata

Definition 5.1 (Two way finite automata)

Two way deterministic finite automaton is a five–tuple $A = (Q, \Sigma, \delta, q_0, F)$, where

Q is a finite set of states,

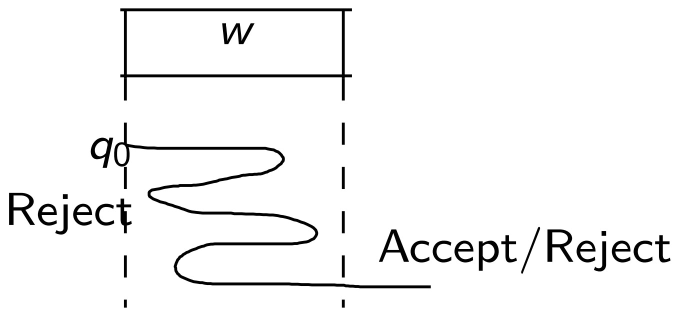
Σ is a finite set of input symbols

transition function δ is a mapping $Q \times \Sigma \to Q \times \{-1, 1\}$ extended by head transitions

$q_0 \in Q$ initial state

a set of accepting states $F \subseteq Q$.

- We may represent it by a graph or a table.

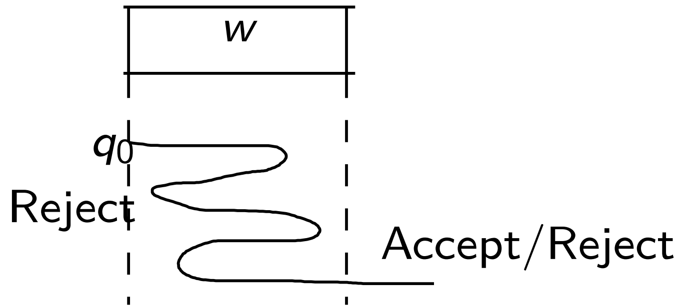


Two–way DFA computation

Definition 5.2 (Two–way DFA computation)

A string w is **accepted by the two–way DFA**, iff:

- computation started in the initial state at the left–most symbol of w
- the first transition from w to the right was in an accepting state
- the computation is not defined outside the word w (computation ends without accepting w).



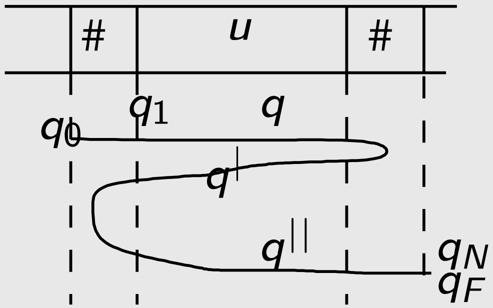
- We may add special end–symbols $\# \notin \Sigma$ to any word
- If $L(A) = \{\#w\# | w \in L \subseteq \Sigma^*\}$ is regular, then also L is regular
- $L = \partial_{\#}\partial^R_{\#}(L(A) \cap \#\Sigma^*\#)$

Two–way automaton example

Example 5.1 (Two–way automaton example)

Let $A = (Q, \Sigma, \delta, q_1, F)$. We define a two–way DFA $B = (Q \cup Q^{|} \cup Q^{||} \cup \{q_0, q_N, q_F\}, \Sigma, \delta^{|}, q_0, \{q_F\})$ accepting the language $L(B) = \{\#u\# | uu \in L(A)\}$ (it is neither left nor right quotient!):

$\delta^{|}$	$x \in \Sigma$	$\#$	remark
q_0	$q_N, -1$	$q_1, +1$	q_1 is starting in A
q	$p, +1$	$q^{|}, -1$	$p = \delta(q, x)$
$q^{|}$	$q^{|}, -1$	$q^{||}, +1$	
$q^{||}$	$p^{||}, +1$	$q_F, +1$	$q \in F, p = \delta(q, x)$
$q^{||}$	$p^{||}, +1$	$q_N, +1$	$q \notin F, p = \delta(q, x)$
q_N	$q_N, +1$	$q_N, +1$	
q_F	$q_N, +1$	$q_N, +1$	



Theorem 5.1

Languages accepted by two–way DFA are exactly regular languages.

Two–way DFA and Regular Languages

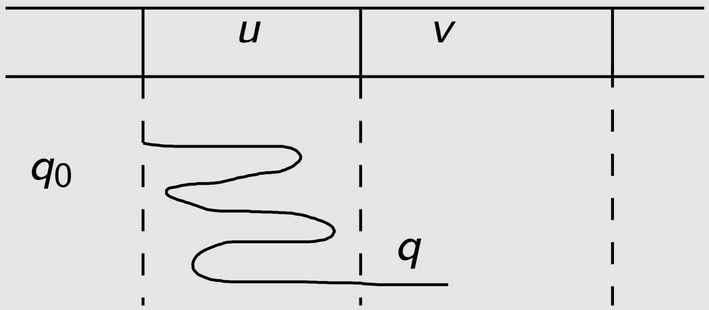
Proof: DFA → two–way DFA

- To a DFA we add the move of the head to the right
- $A = (Q, \Sigma, \delta, q_0, F) \to 2A = (Q, \Sigma, \delta^{|}, q_0, F)$, where $\delta^{|}(q, x) = (\delta(q, x), +1)$. □

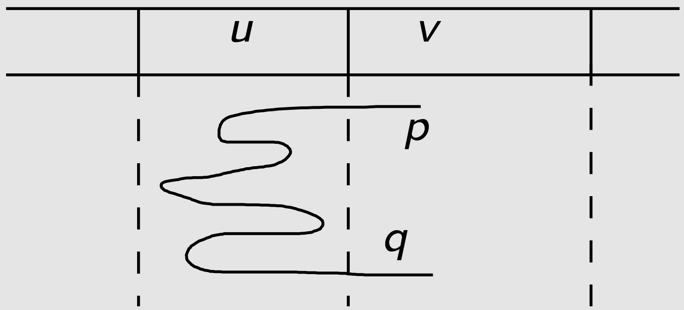
- For the other direction, we need introduction.

The influence of $u \in \Sigma^*$ on the computation over $v \in \Sigma^*$

- the first time we leave u to the right



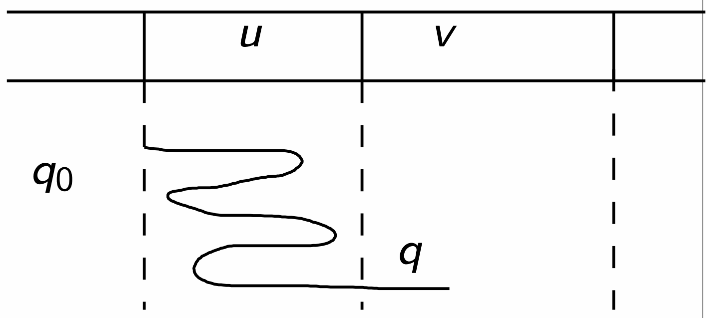
- we leave v to the left and return back v



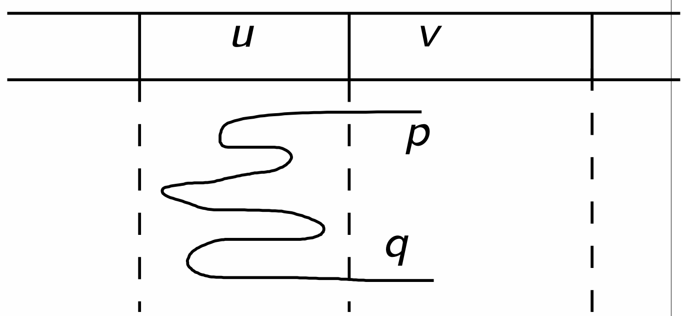
Function f_u describing computation two–way DFA over u

Algorithm: Function f_u describing computation two–way DFA over u

We define $f_u : Q \cup \{q_0^{|}\} \to Q \cup \{0\}$



- $f_u(q_0^{|})$ the state of the first transition to the right in case the computation begins left in the state q_0,



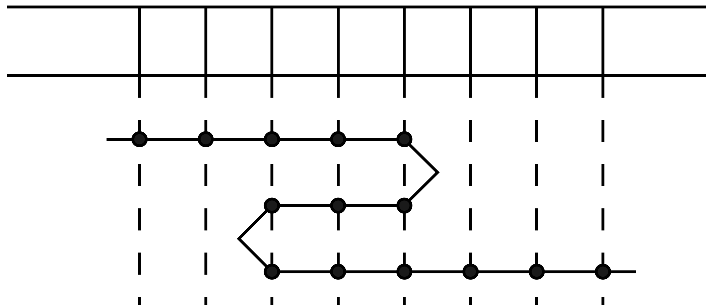
- $f_u(p); p \in Q$ the state of the right transition in case the computation begins right in p
- the symbol 0 denotes failure (a cycle or the head moves left from the initial symbol)
- We define similarity $\sim$ on strings: $u \sim w \Leftrightarrow_{def} f_u = f_w$,
    - strings are similar iff they define identical function f

Languages recognized by two–way DFA are regular

Similarity $\sim$ is a right congruence with a finite index.
According to Myhill–Nerode theorem is the language $L(A)$ regular.

Constructive proof

- We need the left–right movement transcript to a linear computation.
- we are interested in accepting computations only.
- We focus on transitions in cuts between input symbols



Observations:

- The direction of movement repeats (right, left)
- the first and the last transitions are to the right
- automaton is deterministic, any accepting computation is without cycles
- the first and the last cut contain only one state.

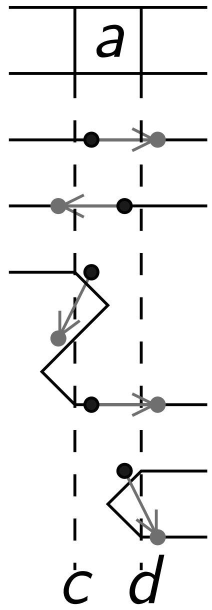
Algorithm: 2DFA → NFA

- Find all possible cuts – state sequences (its a finite number).
- Define non–deterministic transition between cuts according to the input symbol.
- We re–construct the computation by composing cuts like a puzzle.

Algorithm: Formal reduction two–way DFA to NFA

Let $A = (Q, \Sigma, \delta, q_0, F)$be a two–way DFA. We define an equivalent NFA $B = (Q^{|}, \Sigma, \delta^{|}, (q_0), F^{|})$ as follows:

- $Q^{|}$ all possible correct transition sequences
  - sequences of states $(q^1, \ldots, q^k)$; $q^i \in Q$
  - with an odd length $(k = 2m + 1)$
  - no state repeats at odd nor at even position $(\forall i \neq j)\ (q^{2i} \neq q^{2j})\&(\forall i \neq j)\ (q^{2i+1} \neq q^{2j+1})$
- $F^{|} = \{(q) | q \in F\}$ sequences of the length 1
- $\delta^{|}(c, a) = \{d | d \in Q^{|} \& c \xrightarrow{a} d$ is a locally consistent transition for $a\}$
  - there is a bijection: $h : c_{odd} \cup d_{even} \rightarrow c_{even} \cup d_{odd}$ so that:
  - for $h(q) \in c_{even}$ is $(h(q), -1) = \delta(q, a)$
  - for $h(q) \in d_{odd}$ is $(h(q), +1) = \delta(q, a)$



$L(A) = L(B)$

Trajektory two–way DFA A corresponds to cuts in NFA B, therefore $L(A) = L(B)$.

Example Reduction Two–way DFA to NFA

- Let us have two–way DFA:

	a	b
$\rightarrow p$	p,+1	q,+1
$*q$	q,+1	r,-1
r	p,+1	r,-1

Possible cuts and their transitions

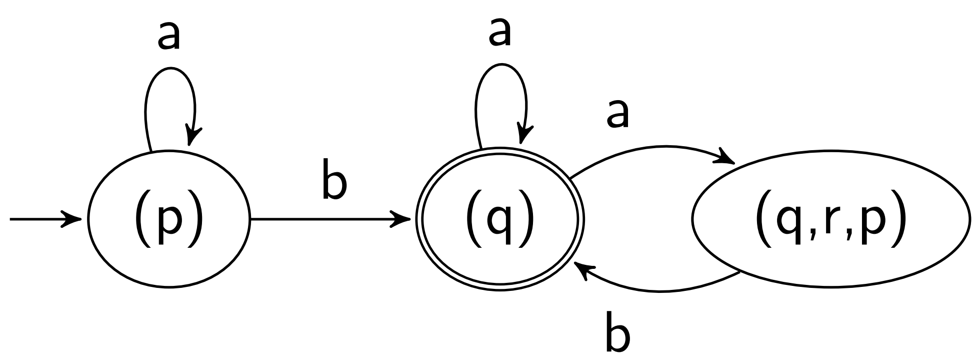
- leftwards only r – all even positions r, that means only one even position
- possible cuts: $(p), (q), (p, r, q), (q, r, p)$.

	a	b
$\rightarrow (p)$	(p)	(q)
$*(q)$	(q),(q,r,p)	
(p,r,q)		
(q,r,p)		(q)

Non–accepting computation example:

a	a	b	a	a	b	a	a	b	b
p	p	p	q	q	q				
					r				
					p	q	q	q	
								r	
								p	q
								r	r
								p	q
								.	.

Resulting NFA:



Automata with the output

Definition 5.3 (Moore machine)

Moore machine is a sixtuple $A = (Q, \Sigma, Y, \delta, \mu, q_0)$ consisting of

Q non–empty set of states

Σ finite nonempty set of symbols (input alphabet)

Y finite nonempty set of symbols (**output alphabet**)

δ a mapping $Q \times \Sigma \to Q$ (transition function)

μ a mapping $Q \to Y$ (**output function**)

$q_0 \in Q$ (initial state)

- the output function may imitate final states
  - $F \subseteq Q$ may be replaced by output function $\mu : Q \to \{0, 1\}$ as follows:

    $\mu(q) = 0$ if $q \notin F$,
    $\mu(q) = 1$ if $q \in F$.

Moore Machine Example

Example 5.2 (Tennis Game Score)

A machine calculates the tenis score.

- Input alphabet: ID of the player who scored a point
- Output alphabet & states: the score ($Q = Y$ and $\mu(q) = q$)

State/output	A	B
00:00	15:00	00:15
15:00	30:00	15:15
15:15	30:15	15:30
00:15	15:15	00:30
30:00	40:00	30:15
30:15	40:15	30:30
30:30	40:30	30:40
15:30	30:30	15:40
00:30	15:30	00:40
40:00	A	40:15
40:15	A	40:30
40:30	A	deuce
30:40	deuce	B
15:40	30:40	B
00:40	15:00	B
deuce	A:40	40:B
A:40	A	deuce
40:B	deuce	B
A	15:00	00:15
B	15:00	00:15

Mealy machine

Definition 5.4 (Mealy machine)

Mealy machine is a six–tuple $A = (Q, \Sigma, Y, \delta, \lambda_M, q_0)$ consisting of:

- Q non–empty set of states
- Σ finite nonempty set of symbols (input alphabet)
- Y finite nonempty set of symbols (output alphabet)
- δ a mapping $Q \times \Sigma \to Q$ (transition function)
- λ_M a mapping $Q \times \Sigma \to Y$ (output function)
- $q_0 \in Q$ (initial state)

- The output is determined by a state and the input symbol
  - Mealy machine is more general then Moore
  - The output function may be replaced as follows

$$\forall x \in \Sigma \ \lambda_M(q, x) = \mu(q)$$

$$\text{or } \forall x \in \Sigma \ \lambda_M(q, x) = \mu(\delta(q, x))$$

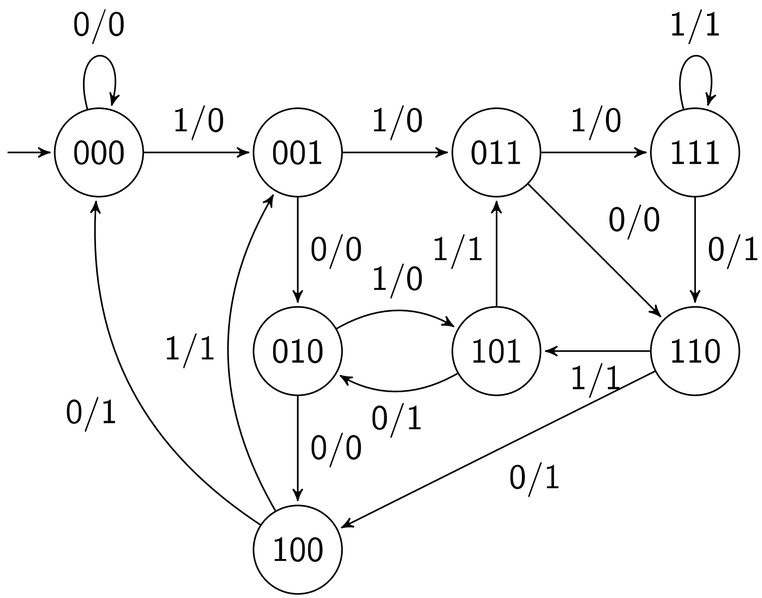
Mealy Machine Example

Example 5.3 (Mealy Machine)

The automaton for integer division of the input by 8 (the reminder is discarded).

- Three bit move to the left
- we need to remember last three bits
- three–bit dynamic memory.

State\symbol	0	1
→000	000/0	001/0
001	010/0	011/0
010	100/0	101/0
011	110/0	111/0
100	000/1	001/1
101	010/1	011/1
110	100/1	101/1
111	110/1	111/1

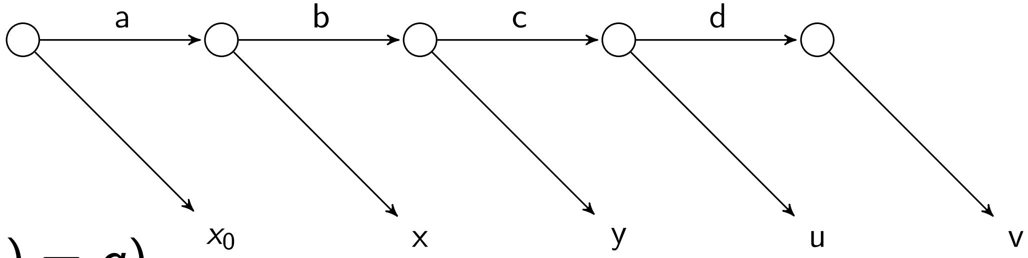


- After three steps calculates properly non–regarding the initial state.

Extended Output Function

for any word in the input alphabet Σ^* $\rightarrow$ we get a word in the output alphabet Y^*

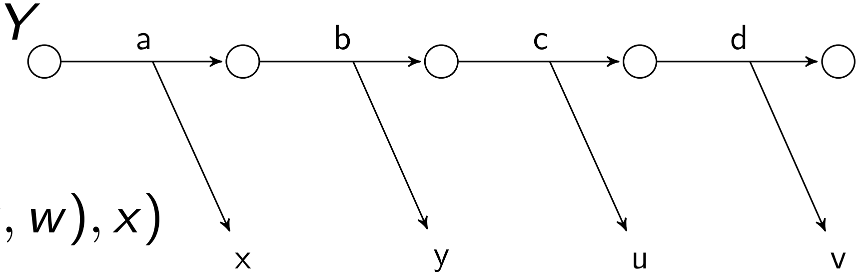
Moore machine



output function $\mu : Q \rightarrow Y$
$\mu^* : Q \times \Sigma^* \rightarrow Y^*$
$\mu^*(q, \lambda) = \lambda$ (sometimes $\mu^*(q, \lambda) = q$)
$\mu^*(q, wx) = \mu^*(q, w).\mu(\delta^*(q, wx))$

Example: μ^*(00:00,AABA)=(00:00 .) 15:00 . 30:00 . 30:15 . 40:15

Mealy machine



output function $\lambda_M : Q \times \Sigma \rightarrow Y$
$\lambda_M^* : Q \times \Sigma^* \rightarrow Y^*$
$\lambda_M^*(q, \lambda) = \lambda$
$\lambda_M^*(q, wx) = \lambda_M^*(q, w).\lambda_M(\delta^*(q, w), x)$

Example: λ_M^*(000,1101010)=0001101

Lemma (Moore and Mealy Machines Reductions)

- *For any Moore machine there exists a Mealy machine mapping each input word to the same output word.*
- *For any Mealy machine there exists a Moore machine mapping each input word to the same output word.*

Proof.

$\Rightarrow$ Mealy machine $B = (Q, \Sigma, Y, \delta, \lambda_M, q_0)$ where $\lambda_M(q, x) = \mu(\delta(q, x))$

$\Leftarrow$ We define states of the Moore machine $Q \times Y$,
$\delta^{|}([q, y], x) = [\delta(q, x), \lambda(q, x)]), \mu([q, y]) = y$.

$\square$