CSC 320 Foundations of Computer Science

Lecture 3

Instructor: Dr. Ulrike Stege

Territory Acknowledgement

We acknowledge and respect the lək̓ʷəŋən peoples on whose traditional territory the university stands and the Songhees, Esquimalt and W̱SÁNEĆ peoples whose historical relationships with the land continue to this day.

This meeting will be recorded

"Please be aware our sessions are being screen-recorded to allow students who are not able to attend to watch later and will be posted in Brightspace."

Deadlines; Assessment

10%	25%	25%	40%
Quizzes	Assignments	Midterms	Final Exam
Quiz 1-8: 1% each Quiz 9: 2%	Assignment 1-5: 5% each	Midterm 1: 10% Midterm 2: 15%	

May

S	M	T	W	T	F	S
			3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

June

S	M	T	W	T	F	S
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	1
2	3	4	5	6	7	8

July

S	M	T	W	T	F	S
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	

Timed quizzes (~30 min)
Review before starting quiz

Last time ….

- Terminology
- The set that contains all languages is uncountable
- Definition of Deterministic Finite Automaton (DFA)

Last time: Terminology, including

- Alphabet Σ: finite set
- Σ^***: set of all strings** over an alphabet Σ
  - Note: empty string $\epsilon \in \Sigma^*$
- Language: subset of Σ^*
- Concatenation of languages: $L_1 L_2 = \{w \in \Sigma^* \mid w = xy$ for some $x \in L_1$ and $y \in L_2\}$

Last time: Terminology, including

- **Kleene star** L^* of a language L: set of all strings obtained by **concatenating zero or more strings** from L

$$L^* = \{w \in \Sigma^* \mid w = w_1 w_2 \ldots w_k,\ k \geq 0 \text{ and } w_i \in L \text{ for } 1 \leq i \leq k\}$$

- **Closure** L^+ of L**:** $L^+ = LL^*$
    - smallest language that includes L and all strings that are concatenations of strings in L

L^+ vs L^*

- $\Sigma = \{a, b, c\}$
- $L_1 = \{a, bc\}$
- $L_1^* = ?$
- $L_2^+ = ?$

http://etc.ch/qn45

$\mathscr{P}(\mathbb{N})$ is uncountable

Proof by contradiction & diagonalization

- **Assume** that $\mathscr{P}(\mathbb{N})$ **is countable**. Then $\mathscr{P}(\mathbb{N})$ is countably infinite

- We list every subset of $\mathbb{N}$ as $S_0, S_1, S_2, \ldots$ s.t.
  - every subset of $\mathbb{N}$ is equal to a subset S_i for some i

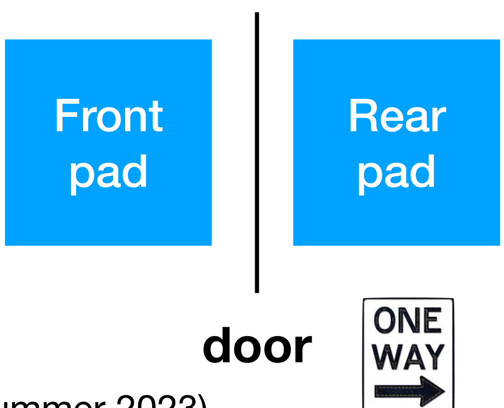
Goal: determine a subset that should be on the list but is not

- Consider subset $D \subseteq \mathbb{N}$: $D = \{i \in \mathbb{N} \mid i \notin S_i\}$
  - For each $j \in \mathbb{N}$, $j \in D$ if and only if $j \notin S_j$
  - Since $D \subseteq \mathbb{N}$: D is on above list & there is some $j_0 \in \mathbb{N}$ with $S_{j_0} = D$
    - If $j_0 \in D$ then $j_0 \notin S_{j_0} = D$
    - If $j_0 \notin D$ then $j_0 \in S_{j_0} = D$
  - That is: $j_0 \in D$ if and only if $j_0 \notin D$

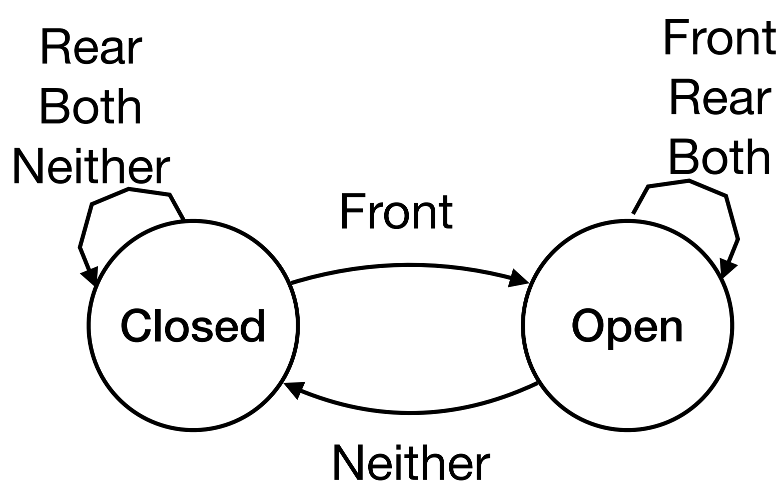
Finite Automata

- Finite automata are all around us! Example: electromechanical devices
  - Controller of an automatic door

Automatic entrance door view from above



Controller



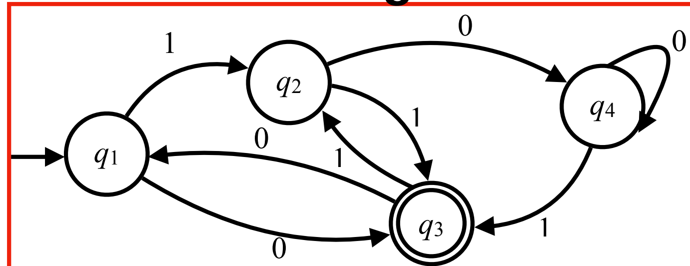
State diagram

Formal Definition
Deterministic Finite Automaton

A **deterministic finite automaton (DFA)** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ with

1. Q is a finite set called the **states**
2. Σ is a finite set called the **alphabet**
3. Function δ: $Q \times \Sigma \to Q$ is the **transition function**
4. $q_0 \in Q$ is the **start state**
5. $F \subseteq Q$ is the **set of accept** (or **final**) **states**

State diagram



$(\{q_1,q_2,q_3,q_4\}, \{0,1\}, \delta, q_1, \{q_3\})$ with δ: $Q \times \Sigma \to Q$ defined by

δ	0	1
q_1	q_3	q_2
q_2	q_4	q_3
q_3	q_1	q_2
q_4	q_4	q_3

Transition table

Today

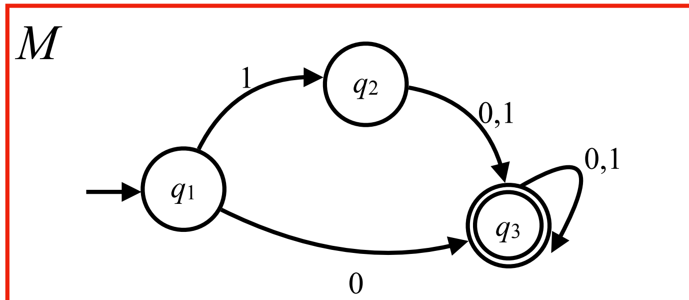
- DFA: language of a DFA, computation of a DFA
- Regular languages
- Closure properties of regular languages

Language of a Deterministic Finite Automaton

- Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA and A be the set of all strings that M accepts
  - A is called the **language of machine** M
  - $L(M) = A$
  - M **recognizes** language A
- Note: a machine that accepts no string recognizes **empty language** $\varnothing$

Example:
Language $L(M)$ of DFA M

- $L(M) = \{w | w \in \Sigma^*$ of length at least 1 where: if w starts with symbol 1 then w is of length at least $2\}$



$(\{q_1,q_2,q_3,q_4\}, \{0,1\}, \delta, q_1, \{q_3\})$ with $\delta: Q \times \Sigma \to Q$ defined by

δ	0	1
q_1	q_3	q_2
q_2	q_3	q_3
q_3	q_3	q_3

DFA: Formal Definition of Computation

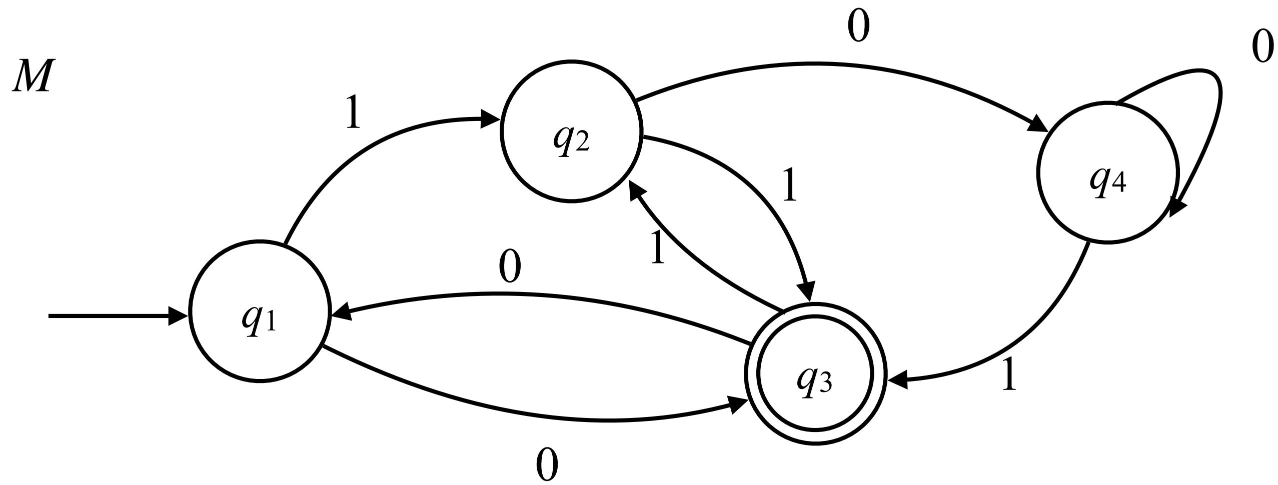
- Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA and let $w = w_1 w_2 \ldots w_n$ be a string over Σ. Then M **accepts** w if there is a sequence of states $r_0, r_1, r_2, \ldots, r_n$ in Q such that

    1. $r_0 = q_0$
    2. $\delta(r_i, w_{i+1}) = r_{i+1}$
    3. $r_n \in F$

M recognizes language L if $L = L(M) = \{w \in \Sigma^* \mid M \text{ accepts } w\}$

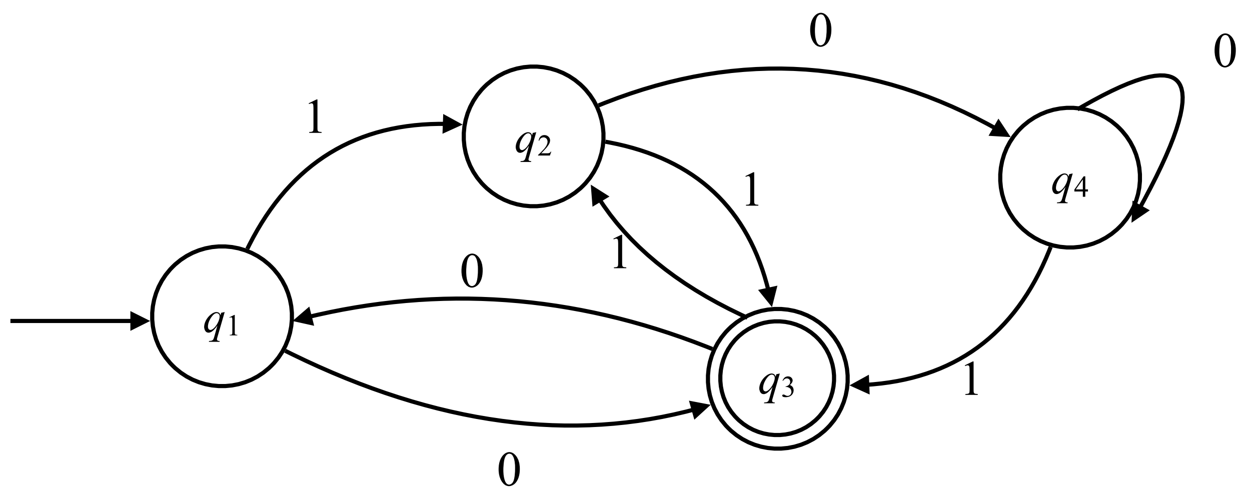
- A language L is called a **regular language** if there exists a DFA that recognizes L

Computation of DFA—Example Sequence of states for $w = 001101$



- $q_1, q_3, q_1, q_2, q_3, q_1, q_2$
- Since q_2 is not a final state w is *not* accepted
- $w \notin L(M)$

Computation of DFA—Example
Sequence of states for $w = 0011011$



- $q_1,\ q_3,\ q_1,\ q_2,\ q_3,\ q_1,\ q_2,\ q_3$
- Since q_3 is a final state w is accepted
- $w \in L(M)$

Regular languages

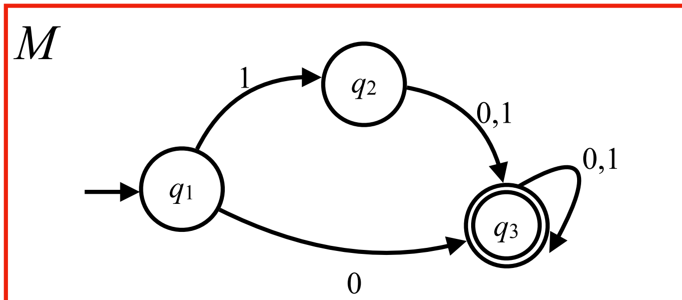
- Given DFA M, language $L(M)$ is **exactly the set of all strings that M accepts**

- $L(M)$ is a regular language

Example: Language of DFA M

- $L(M) = \{w | w \in \Sigma^*$ of length at least 1 where: if w starts with symbol 1 then w is of length at least $2\}$

- $L(M)$ is a regular language



$(\{q_1, q_2, q_3, q_4\}, \{0,1\}, \delta, q_1, \{q_3\})$ with $\delta: Q \times \Sigma \rightarrow Q$ defined by

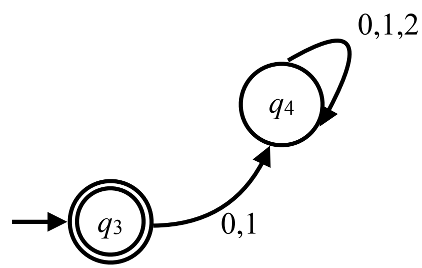
δ	0	1
q_1	q_3	q_2
q_2	q_3	q_3
q_3	q_3	q_3

Regular languages—what we know so far

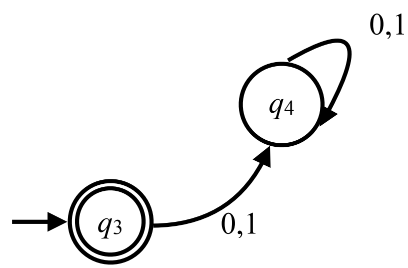
A language L is called a **regular language** if there exists a deterministic finite automaton that recognizes L

The set of languages that are recognized by the set of all DFAs, the **regular languages**, is a **subset of the set of all languages**

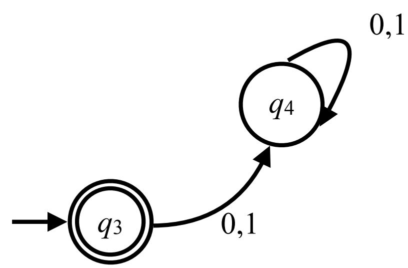
Is this a DFA?



Is this a DFA?



What is $L(M)$?



Regular Languages: Closure Properties

- Definition: closed
- Operations: union, intersection, concatenation

Closure Properties for Sets

- A **set** is **closed** under some operation if applying that operation to elements of the set returns another element of the set

- If a certain operation applied to **any language** in a certain **class of languages** (eg, the class of regular languages) produces a result that is also in that same class, then the language class (eg, the class of regular languages) is **closed under this operation**

Regular Languages: Closure Properties

- We show: regular languages are closed under
  - Union
  - Intersection
  - Concatenation

Regular Languages: Closure Properties

Theorem. If L_1 and L_2 are regular languages over alphabet Σ then $L_1 \cup L_2$ is a regular language

In other words: *The class of regular languages is closed under the union operation*

Proof. Since L_1 and L_2 are regular languages there exist deterministic finite automata M_1 and M_2 with $L_1 = L(M_1)$ and $L_2 = L(M_2)$

Idea. Construct DFA M that accepts exactly the strings accepted by M_1 and the strings accepted by M_2

Regular Languages: Closure Properties

Proof. Since L_1 and L_2 are regular languages there exist DFA M_1 and M_2 with $L_1 = L(M_1)$ and $L_2 = L(M_2)$

Idea: Construct DFA M that accepts exactly the strings accepted by M_1 and the strings accepted by M_2

We do this by simulating both machines concurrently, and accepting if (at least) one of them accepts

Proof continued

Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ be DFAs.
We construct DFA $M = (Q, \Sigma, \delta, q_0, F)$ as follows

- $Q = \{(r_1,r_2) \mid r_1 \in Q_1,\ r_2 \in Q_2\}$

- For each $(r_1,r_2) \in Q$ and each $a \in \Sigma$ define transition function δ: $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$

- $q_0 = (q_1,q_2)$

- $F = \{(r_1,r_2) \mid r_1 \in F_1$ or $r_2 \in F_2\}$

M **recognizes** $L_1 \cup L_2$

Why does M recognize $L_1 \cup L_2$?

- Let's first look at an example

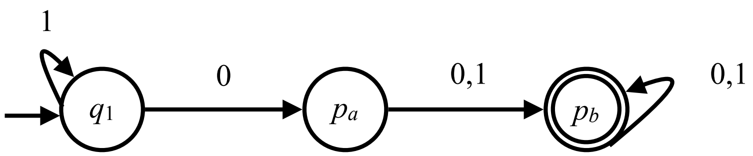
Example

$\Sigma = \{0,1\}$

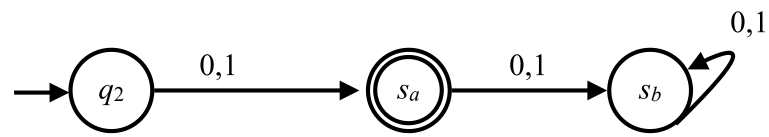
L_1: set of all strings that, after a possible prefix of 1s, consist of at least one 0 followed by at least one symbol

L_2: set of all strings of length at exactly 1

M_1



M_2



Transition table for M_1

	Q_1	0	1
start	q_1	p_a	q_1
	p_a	$\boldsymbol{p_b}$	$\boldsymbol{p_b}$
	$\boldsymbol{p_b}$	$\boldsymbol{p_b}$	$\boldsymbol{p_b}$

Transition table for M_2

	Q_2	0	1
start	q_2	$\boldsymbol{s_a}$	$\boldsymbol{s_a}$
	$\boldsymbol{s_a}$	s_b	s_b
	s_b	s_b	s_b

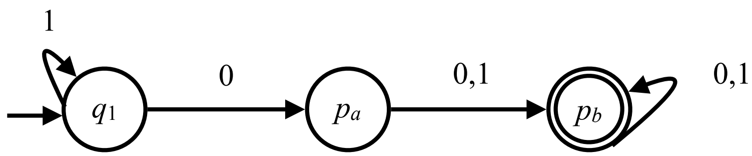
Example

$\Sigma = \{0,1\}$

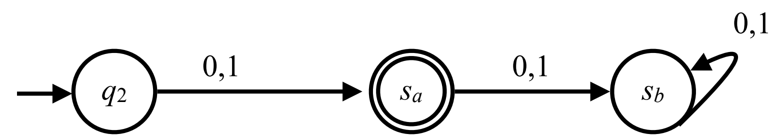
L_1: set of all strings that, after a possible prefix of 1s, consist of at least one 0 followed by at least one symbol

L_2: set of all strings of length at exactly 1

M_1



M_2



Transition table for M

	Q	0	1
start	(q_1,q_2)	(p_a,s_a)	(q_1,s_a)
	(q_1,s_a)	(p_a,s_b)	(q_1,s_b)
	(q_1,s_b)	(p_a,s_b)	(q_1,s_b)
	(p_a,q_2)	(p_b,s_a)	(p_b,s_a)
	(p_a,s_a)	(p_b,s_b)	(p_b,s_b)
	(p_a,s_b)	(p_b,s_b)	(p_b,s_b)
	(p_b,q_2)	(p_b,s_a)	(p_b,s_a)
	(p_b,s_a)	(p_b,s_b)	(p_b,s_b)
	(p_b,s_b)	(p_b,s_b)	(p_b,s_b)

- $Q = \{(r_1,r_2) \mid r_1 \in Q_1,\ r_2 \in Q_2\}$
- $q_0 = (q_1,q_2)$
- For each $(r_1,r_2) \in Q$ & each $a \in \Sigma$

$$\delta((r_1,r_2),a) = (\delta_1(r_1,a), \delta_2(r_2,a))$$

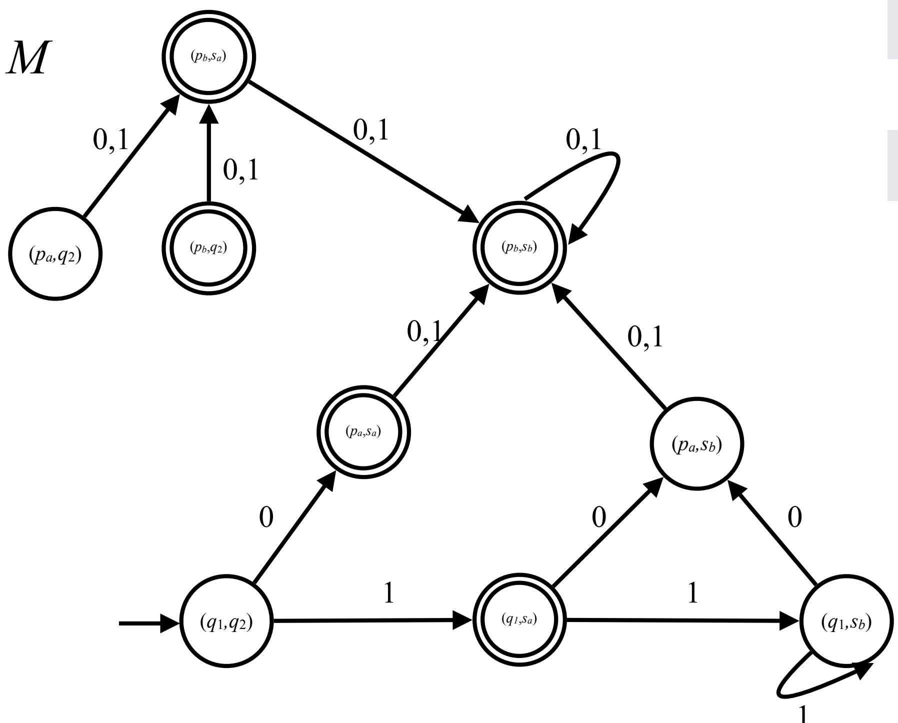
Transitions simulate both DFAs

Example (continued)

- $F = \{(r_1,r_2) \mid r_1 \in F_1$ or $r_2 \in F_2\}$

Q	0	1
start (q_1,q_2)	$\boldsymbol{(p_a,s_a)}$	$\boldsymbol{(q_1,s_a)}$
$\boldsymbol{(q_1,s_a)}$	(p_a,s_b)	(q_1,s_b)
(q_1,s_b)	(p_a,s_b)	(q_1,s_b)
(p_a,q_2)	$\boldsymbol{(p_b,s_a)}$	$\boldsymbol{(p_b,s_a)}$
$\boldsymbol{(p_a,s_a)}$	$\boldsymbol{(p_b,s_b)}$	$\boldsymbol{(p_b,s_b)}$
(p_a,s_b)	$\boldsymbol{(p_b,s_b)}$	$\boldsymbol{(p_b,s_b)}$
$\boldsymbol{(p_b,q_2)}$	$\boldsymbol{(p_b,s_a)}$	$\boldsymbol{(p_b,s_a)}$
$\boldsymbol{(p_b,s_a)}$	$\boldsymbol{(p_b,s_b)}$	$\boldsymbol{(p_b,s_b)}$
$\boldsymbol{(p_b,s_b)}$	$\boldsymbol{(p_b,s_b)}$	$\boldsymbol{(p_b,s_b)}$

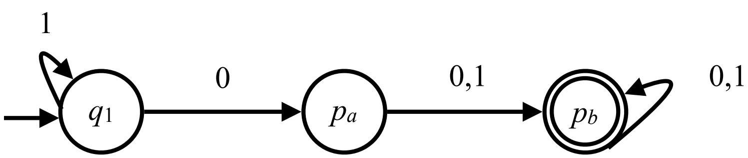
F



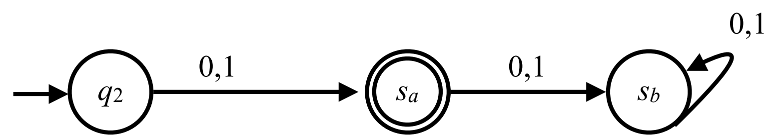
$$L(M) = L_1 \cup L_2$$

Example (continued)

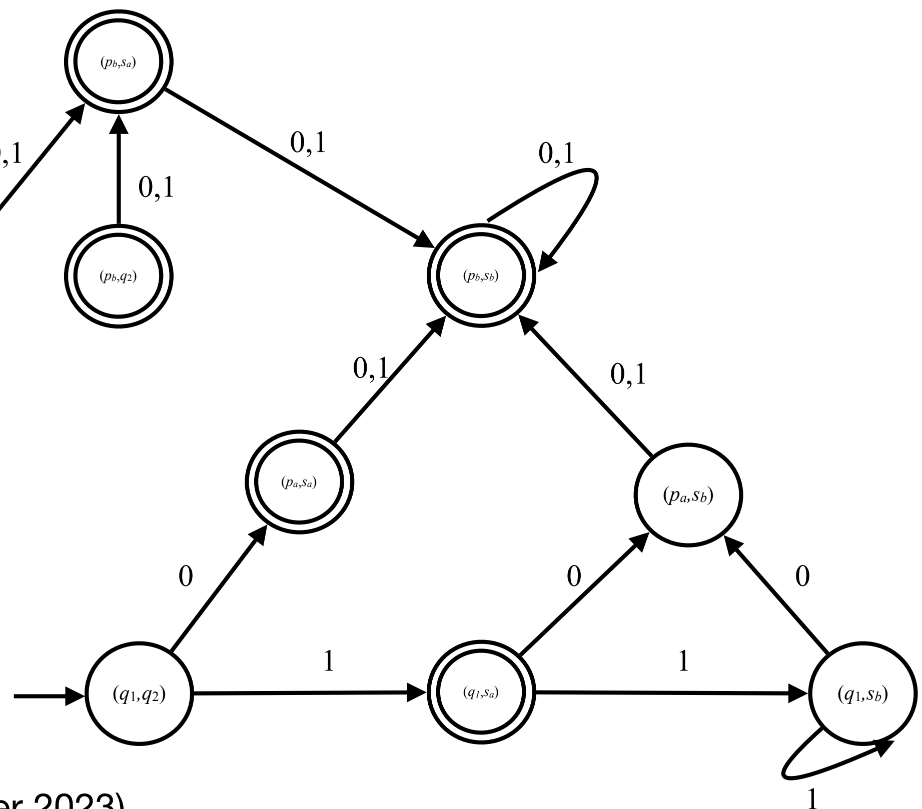
M_1



M_2



M



$L(M) = L_1 \cup L_2$

Why does M recognize $L_1 \cup L_2$?

For $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ be DFAs.
DFA $M = (Q, \Sigma, \delta, q_0, F)$ is

- $Q = \{(r_1,r_2) \mid r_1 \in Q_1,\ r_2 \in Q_2\}$

  M has a state for each pair, where one state is in Q_1 & one is in Q_2

- For each $(r_1,r_2) \in Q$ and each $a \in \Sigma$ define transition function δ: $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$

  Transitions simulate both DFAs

- $q_0 = (q_1,q_2)$

  We start simulating both M_1 and M_2 by a state that simulates both start states

- $F = \{(r_1,r_2) \mid r_1 \in F_1$ or $r_2 \in F_2\}$

  Strings are accepted if at least one DFA accepts

$\boxed{M \text{ recognizes } L_1 \cup L_2}$

Regular Languages: Closure Properties (2)

Theorem. If L_1 and L_2 are regular languages over alphabet Σ then $L_1 \cap L_2$ is a regular language

Proof. Since L_1 and L_2 are regular languages there exists finite automata M_1 and M_2 with $L_1 = L(M_1)$ and $L_2 = L(M_2)$

Idea. Construct a DFA M that accepts exactly the strings accepted by M_1 and M_2; similar to previous proof but need to pay attention what strings must be accepted by M

Proof

Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$. We construct $M = (Q, \Sigma, \delta, q_0, F)$ as follows

- $Q = \{(r_1,r_2) \mid r_1 \in Q_1,\ r_2 \in Q_2\}$
- For each $(r_1,r_2) \in Q$ and each $a \in \Sigma$:
  $\delta((r_1,r_2), a) = (\delta_1(r_1,a),\delta_2(r_2,a))$
- $q_0 = (q_1,q_2)$
- $F = \{(r_1,r_2) \mid r_1 \in F_1$ **and** $r_2 \in F_2\}$

M **recognizes** $L_1 \cap L_2$

$\square$

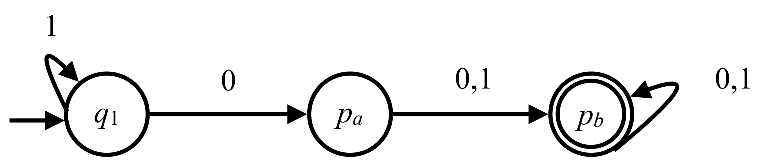
Example

$\Sigma = \{0,1\}$

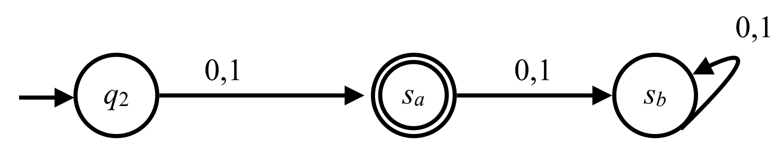
L_1= set of all strings that, after a possible prefix of 1s, consist of at least one 0 followed by at least one symbol

L_2= set of all strings of length at exactly 1

M_1



M_2



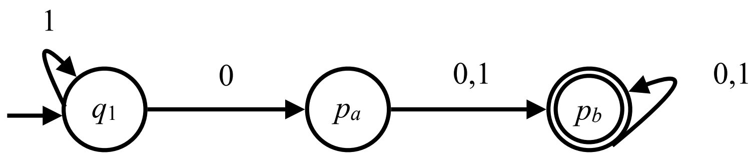
Q	0	1
(q_1,q_2)	(p_a,s_a)	(q_1,s_a)
(q_1,s_a)	(p_a,s_b)	(q_1,s_b)
(q_1,s_b)	(p_a,s_b)	(q_1,s_b)
(p_a,q_2)	(p_b,s_a)	(p_b,s_a)
(p_a,s_a)	(p_b,s_b)	(p_b,s_b)
(p_a,s_b)	(p_b,s_b)	(p_b,s_b)
(p_b,q_2)	(p_b,s_a)	(p_b,s_a)
(p_b,s_a)	(p_b,s_b)	(p_b,s_b)
(p_b,s_b)	(p_b,s_b)	(p_b,s_b)

Example (continued)

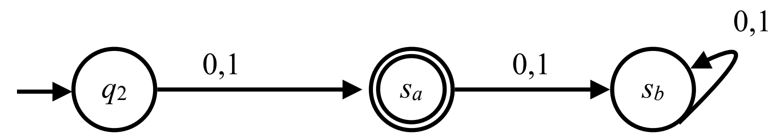
$L(M) = L_1 \cap L_2$

$F = \{(r_1,r_2) \mid r_1 \in F_1$ **and** $r_2 \in F_2\}$

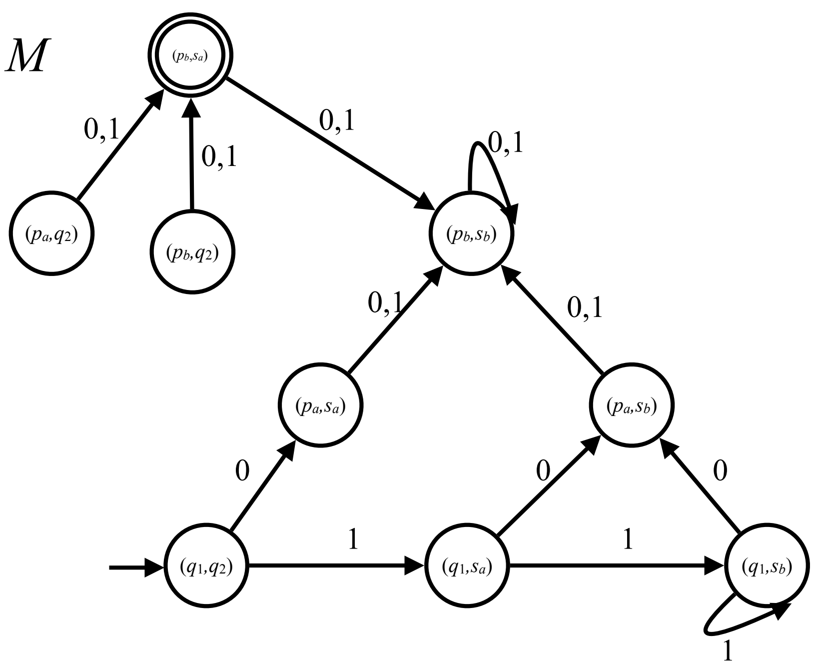
M_1



M_2



M



	Q	0	1
start	(q_1,q_2)	(p_a,s_a)	(q_1,s_a)
	(q_1,s_a)	(p_a,s_b)	(q_1,s_b)
	(q_1,s_b)	(p_a,s_b)	(q_1,s_b)
	(p_a,q_2)	$\boldsymbol{(p_b,s_a)}$	$\boldsymbol{(p_b,s_a)}$
	(p_a,s_a)	(p_b,s_b)	(p_b,s_b)
	(p_a,s_b)	(p_b,s_b)	(p_b,s_b)
	(p_b,q_2)	$\boldsymbol{(p_b,s_a)}$	$\boldsymbol{(p_b,s_a)}$
	$\boldsymbol{(p_b,s_a)}$	(p_b,s_b)	(p_b,s_b)
	(p_b,s_b)	(p_b,s_b)	(p_b,s_b)

F

Regular Languages: Closure Properties (3)

Theorem. If L_1 and L_2 are regular languages over alphabet Σ then $L_1 \, L_2$ is a regular language

Proof. Since L_1 and L_2 are regular languages there exist DFA M_1 and M_2 with $L_1 = L(M_1)$ and $L_2 = L(M_2)$

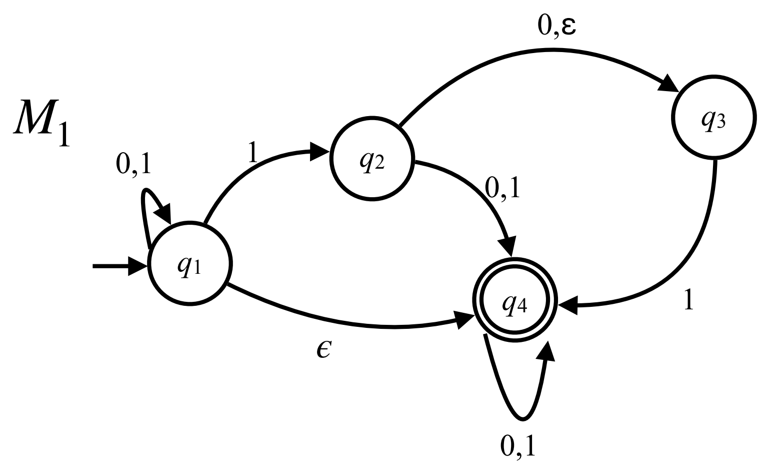
Idea. Construct a finite automaton M that accepts exactly all the strings of which the first part is accepted by M_1 and the second part by M_2

How can this be done?

We introduce: **nondeterminism** & **nondeterministic** finite automata

- Nondeterminism
    - Abstraction that allows to consider extension of ordinary computation
    - Simultaneous execution paths are permitted
    - Strings are accepted if there exists at least one execution path that is an accepting one
- Proving languages closed under concatenation
    1. Prove for nondeterministic fine automata and their corresponding language
    2. Show that nondeterministic finite automata accept the same class of language as deterministic ones, ie regular languages

NFA Example



$M_1 = (\{q_1,q_2,q_3,q_4\}, \{0,1\}, \delta, q_1, \{q_4\})$ with

δ: $Q\times(\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q)$ defined by

δ	0	1	ε
q_1	$\{q_1\}$	$\{q_1,q_2\}$	$\{q_4\}$
q_2	$\{q_3,q_4\}$	$\{q_4\}$	$\{q_3\}$
q_3	$\varnothing$	$\{q_4\}$	$\varnothing$
q_4	$\{q_4\}$	$\{q_4\}$	$\varnothing$

Note: state diagram omits transitions into $\varnothing$

Is there a string in Σ^* that is not accepted by M_1?