CSC 320
Foundations of Computer Science

Lecture 4

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%

Quizzes

Quiz 1-8: 1% each
Quiz 9: 2%

25%

Assignments

Assignment 1-5: 5% each

25%

Midterms

Midterm 1: 10%
Midterm 2: 15%

40%

Final Exam

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

What about

- Reading Assignments?

Last time ….

- DFA, regular languages, closure properties of regular languages

Today …

- NFAs
- Equivalence of NFAs and DFAs
- Regular languages are closed under concatenation
- Regular languages

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

Proof

Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ 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** $L_1 \cup L_2$

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** $L_1 \cap L_2$

□

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 $L_2 = L(M_2)$

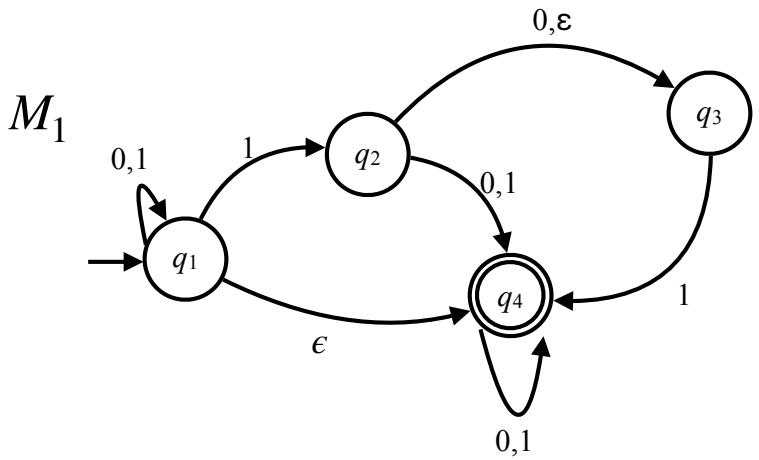
Idea. Construct a finite automaton M that accepts exactly all the strings of which the first part is accepted by M_1 and the second part by M_2

How can this be done?

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: $Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \mathcal{P}(Q)$ 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 ∅

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

Nondeterministic Finite Automata (NFA)

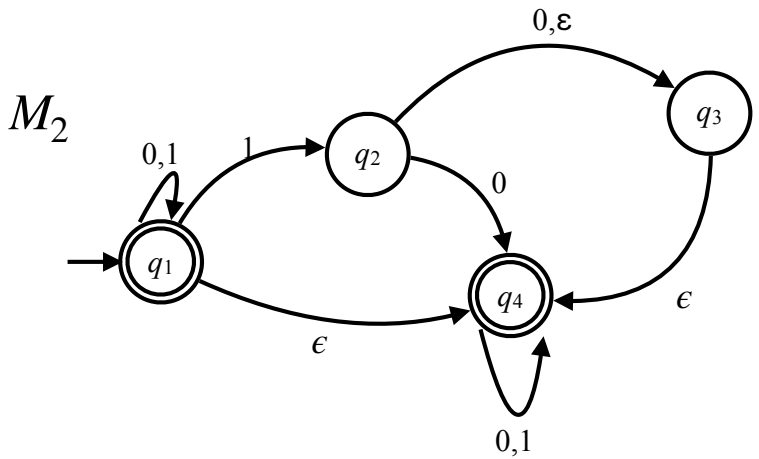
- Transitions go from states to **sets of states**
  - Starting from a state q and reading a symbol a can have transitions into more than one state

$$\delta: Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \mathcal{P}(Q)$$

Powerset of Q

- We allow **empty transitions** (ϵ-transitions) $\xrightarrow{\epsilon}$
  - Do not read any symbol from the input

NFA Example (2)



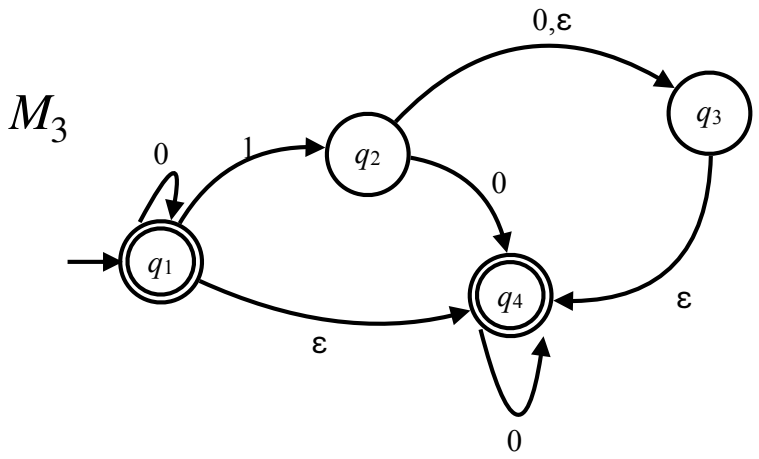
$M_2 = (\{q_1,q_2,q_3,q_4\}, \{0,1\}, \delta, q_1, \{q_1, q_4\})$

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

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

Is $L(M_1) = L(M_2)$?

NFA Example (3)



$M_3 = (\{q_1,q_2,q_3,q_4\}, \{0,1\}, \delta, q_1, \{q_1, q_4\})$

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

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

Is $L(M_1) = L(M_3)$?

Formal Definition
Nondeterministic Finite Automaton

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

1. Q is a finite set of states
2. Σ is an alphabet
3. Function δ: $Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$ 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

NFA: Computation

- Let $M = (Q, \Sigma, \delta, q_0, F)$ be a NFA and $w = w_1 w_2 \ldots w_n$ a string over Σ. Then M **accepts** w if we can write $w = y_1 y_2 \ldots y_m$ with $y_i \in \Sigma \cup \{\epsilon\}$ and there is a sequence of states $r_0, r_2, \ldots, r_m$, $r_i \in Q$, such that

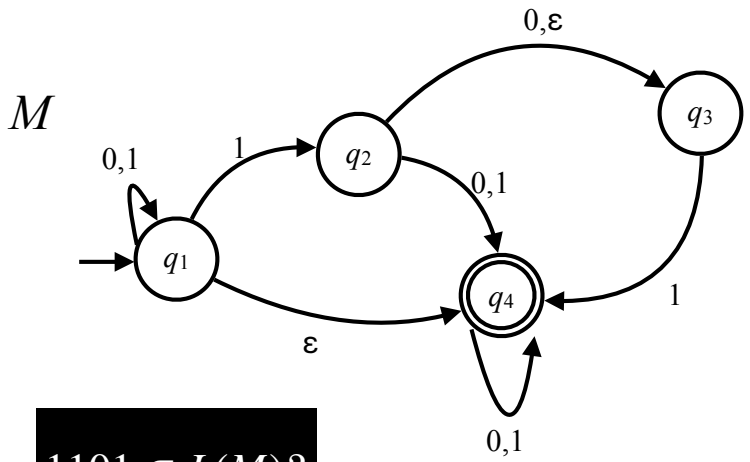
  1. $r_0 = q_0$
  2. $r_{i+1} \in \delta(r_i, y_{i+1})$
  3. $r_m \in F$

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

- If a machine M that does not accept any string then $L(M) = \emptyset$

Same as DFA

NFA Example



$1101 \in L(M)$?

What about string 00101? How many different state sequences can you find? How many show acceptance?

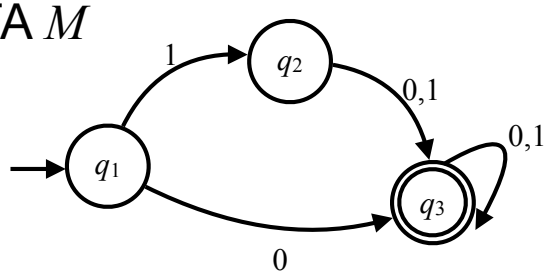
- Input $w = 1101$ is accepted by M, that is $w \in L(M)$
  - Evidence: rewriting $w = 1\varepsilon101$ and state sequence $q_1, q_2, q_3, q_4, q_4, q_4$
  - Also q_1, q_1, q_4, q_4, q_4 is a sequence of states for w (for 1101), and for the same rewriting: $w = 1\epsilon101$
- Note that choosing sequence q_1, q_1, q_1, q_1 for w (for 1101) does not yield acceptance, but $w \in L(M)$

NFAs and DFAs

- A DFA can be considered a special case of NFA
- Formal definition of transition function is different

Example

DFA M

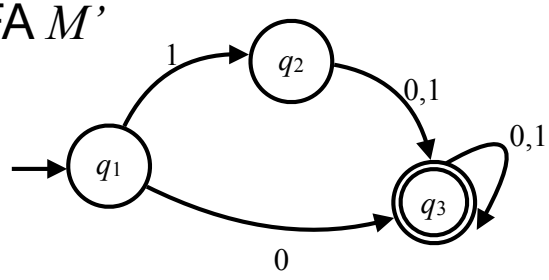


$(\{q_1,q_2,q_3,q_4\}, \{0,1\}, \delta, q_1, \{q_3\})$ with

δ: $Q \times \Sigma \to Q$ defined by

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

NFA M'

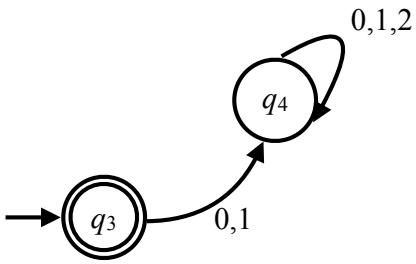


$(\{q_1,q_2,q_3,q_4\}, \{0,1\}, \delta, q_1, \{q_3\})$ with

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

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

Is this an NFA?



NFAs and DFAs

- Definition: Let M_1 and M_2 each be a DFA or NFA. Then we call M_1 and M_2 **equivalent** if $L(M_1) = L(M_2)$

- **Observation:** For every deterministic finite automaton there exists an equivalent nondeterministic finite automaton

Show: For every deterministic finite automaton there exists an equivalent nondeterministic finite automaton

Let $M = (Q, \Sigma, \delta, q_M, F)$ be a DFA. Then we can build NFA $N = (Q', \Sigma, \delta', q_N, F')$ with $L(M) = L(N)$ as follows:

- $Q' := Q$
- $q_N := q_M$
- $F' := F$
- $\delta' : Q' \times (\Sigma \cup \{\epsilon\}) \rightarrow \mathcal{P}(Q')$ with $\delta'(q, a) := \{\delta(q, a)\}$ for all $a \in \Sigma$, $\delta'(q, \epsilon) := \varnothing$

Equivalence of NFAs & DFAs

Theorem: For every NFA there exists an equivalent DFA

Theorem: For every NFA there exists an equivalent DFA

Proof.

Plan: Given NFA $N = (Q, \Sigma, \delta, q_0, F)$,

construct DFA $D = (Q_D, \Sigma, \delta_D, q_D, F_D)$ with $L(N) = L(D)$

For any given string w to be processed, the DFA has to have *a unique sequence* of states that represents *all possible state sequences for* w in the NFA

Idea: Build D such that it simulates the computation of N

Caution Do not miss any possible computation of N in simulation: When defining states Q_D for D, **create a state for *every possible subset* of Q**

Define δ_D for all those states in Q_D and all input symbols

For now: ignore ϵ-transitions in N, that is we assume N does not have any ϵ-transitions; we deal with them later

Building D

- Given NFA $N = (Q, \Sigma, \delta, q_0, F)$
- Build DFA $D = (Q_D, \Sigma, \delta_D, q_D, F_D)$
  - $Q_D := \mathcal{P}(Q)$, $q_D := \{q_0\}$
  - F_D: set of all subsets of Q that contain a final state of F
  - Definition of δ_D
    - Let state $S \in Q_D$ be a state of DFA D and let $a \in \Sigma$
      Recall $S \subseteq Q$
    - $\delta_D(S, a) := \{q \in Q \mid q \in \delta(s, a) \text{ for some } s \in S\}$
  - $F_D := \{S \in Q_D \mid$ there exists a $q \in S$ with $q \in F\}$
    ie $F_D = \{S \in Q_D \mid S \cap F \neq \varnothing\}$

Our construction so far only works for NFAs without ϵ-transitions

- Modify construction to also simulate NFAs with ϵ-transitions

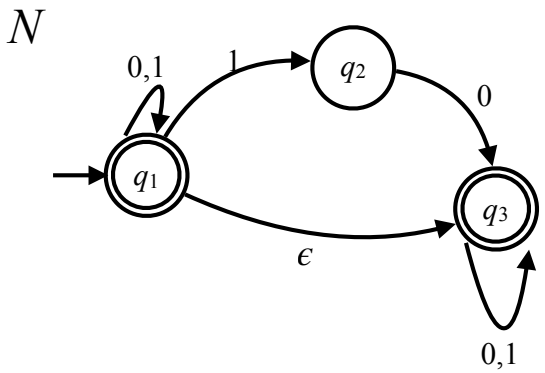
Adding ϵ-Transitions

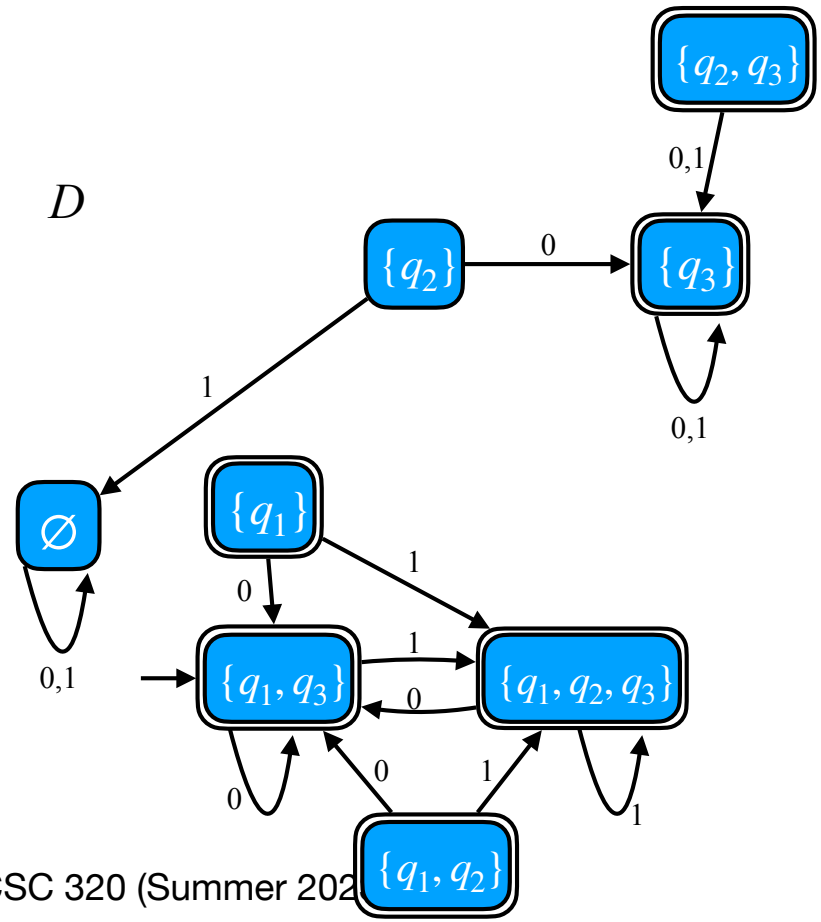
For any $S \in Q_D$ let $E(S) = \{q \mid q$ can be reached from some state in S by traveling 0 or more ϵ-transitions$\}$

Modify D as follows:

- $q_D := E(\{q_0\})$
- $\delta_D(S, a) := \{q \in Q \mid q \in E(\delta(s, a))$ for some $s \in S\}$

Example





	δ_D	0	1
	$\varnothing$	$\varnothing$	$\varnothing$
	$\{q_1\}$	$\{q_1,q_3\}$	$\{q_1,q_2,q_3\}$
	$\{q_2\}$	$\{q_3\}$	$\varnothing$
	$\{q_3\}$	$\{q_3\}$	$\{q_3\}$
	$\{q_1,q_2\}$	$\{q_1,q_3\}$	$\{q_1,q_2,q_3\}$
Start	$\{q_1,q_3\}$	$\{q_1,q_3\}$	$\{q_1,q_2,q_3\}$
	$\{q_2,q_3\}$	$\{q_3\}$	$\{q_3\}$
	$\{q_1,q_2,q_3\}$	$\{q_1,q_3\}$	$\{q_1,q_2,q_3\}$

Summary: Building D

- Given NFA $N = (Q, \Sigma, \delta, q_0, F)$
- Build DFA $D = (Q_D, \Sigma, \delta_D, q_D, F_D)$
  - $Q_D := \mathcal{P}(Q)$, $q_D := E(\{q_0\})$
  - F_D: set of all subsets of Q that contain a final state of F
  - Definition of δ_D
    - Let $S \in Q_D$, $a \in \Sigma$
      - $E(S) = \{q \mid q$ can be reached from some state in S by traveling 0 or more ϵ-transitions$\}$
      - $\delta_D(S, a) := \{q \in Q \mid q \in E(\delta(s, a))$ for some $s \in S\}$
  - $F_D := \{S \in Q_D \mid S \cap F \neq \varnothing\}$

$L(D) = L(N)$

NFAs and DFAs

- Since NFAs and DFAs produce the same set of languages we know:
- The languages recognized by NFAs is exactly the set of regular languages

Regular Languages: Closure Properties

- **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 exists deterministic finite automata M_1 and M_2 with $L_1 = L(M_1)$ and $L_2 = L(M_2)$

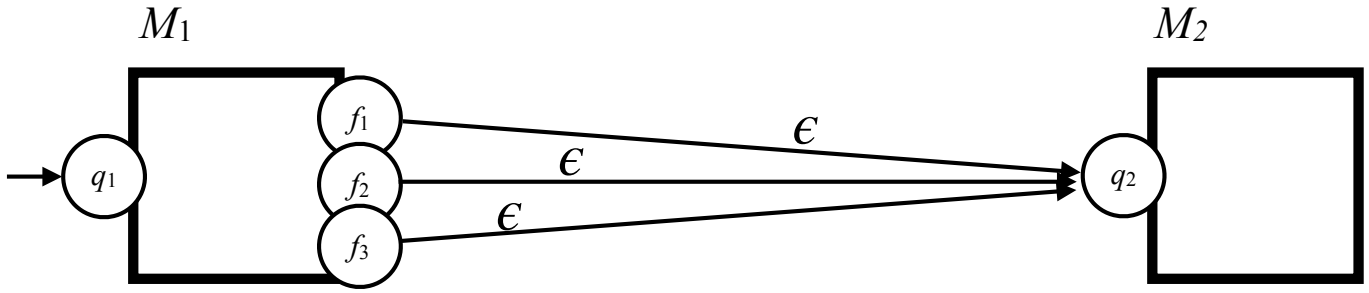
  Idea. Construct a nondeterministic finite automaton M that accepts exactly the strings where the first part is accepted by first M_1 and the second one by M_2

Proof: Regular Languages are closed under concatenation

$M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$

$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

M



M inherits all states from DFAs M_1 and M_1 with

- M's **start state** is M_1's start state
- M's **final states** are M_2's final states
- M's **transitions** consist of:
  - all transitions of M_1 and all transitions of M_2
  - ε-transitions between each state that corresponds to a final state in M_1 and the state the corresponds to M_2's start state

$$M = (Q, \Sigma, \delta, q_1, F_2)$$

We have seen

Given regular languages L_1 and L_2 then

- $L_1 \cup L_2$ is a regular language
- $L_1 \cap L_2$ is a regular language
- $L_1 L_2$ is a regular language

Your turn

- Let L_1, L_2 be regular languages
- Is $L = ((L_1 \cup L_2)L_1) \cap L_2$ a regular language?

Next: Regular Expressions

Regular Expressions

- Wouldn't it be nice if we could describe regular languages shorter?

  Eg: $(0 \cup 1)^*0$ for the language of all strings over the binary alphabet that end with 0

- Examples where regular languages are used in practice (by using regular expressions):
  - Unix: awk, grep
  - Perl
  - Texteditors
  - Design of compilers (lexical analyzer)

Recommended: Read on the web about regular expressions in Unix & Linux

Definition: regular expression

- R is a **regular expression** if R is equal to
  - a, for some $a \in \Sigma$
  - ϵ
  - $\varnothing$
  - $(R_1 \cup R_2)$ where R_1 and R_2 are regular expressions
  - $(R_1 R_2)$ where R_1 and R_2 are regular expression
  - (R_1^*) where R_1 is a regular expression

Conventions: regular expressions

- Parentheses can be omitted
- If no parentheses, order to evaluate is: star, concatenation, union
- $R^+ := RR^*$

Summary: Regular Expressions

- Defined inductively. Regular expressions are:
  - a ($a \in \Sigma$), ϵ, $\varnothing$
  - $(R_1 \cup R_2)$, $(R_1 R_2)$ and $(R_1{}^*)$, where R_1 and R_2 are regular expressions
- Parenthesis can be omitted
  - If no parentheses evaluate in order: star, concatenation, union
- Identities of regular expressions: $R^+ := RR^*$, $R \cup \varnothing = R$, $R\varepsilon = R$

Definition: *Language* recognized by a regular expression

Assume R_1 and R_2 are regular expressions. The **language** $L(R)$ **for regular expression** R is defined as:

- If $R = a$, for some $a \in \Sigma$, then $L(R) = \{a\}$
- If $R = \epsilon$ then $L(R) = \{\epsilon\}$
- If $R = \emptyset$ then $L(R) = \emptyset$
- If $R = (R_1 \cup R_2)$ then $L(R) = L(R_1) \cup L(R_2)$
- If $R = (R_1 R_2)$ then $L(R) = L(R_1)L(R_2)$
- If $R = (R_1^*)$ then $L(R) = L(R_1)^*$

Examples

Let $a, b \in \Sigma$

- $L(a \cup b)$
- $L(a(a \cup b))$
- $L((a \cup b)^*)$
- $L(a(a \cup b)^*)$

Recall

$a \in \Sigma$: $L(a) = \{a\}$

$L(\epsilon) = \epsilon$

$L(\varnothing) = \varnothing$

$L(R_1 \cup R_2) = L(R_1) \cup L(R_2)$

$L(R_1 R_2) = L(R_1)L(R_2)$

$L(R^*) = L(R)^*$

Equivalence of regular languages and the set of languages of regular expressions

Theorem: A language is regular if and only if there exists some regular expression that describes it

Proof consists of two parts

1. If a language is described by a regular expression, then it is regular

2. If a language is regular, then it can be described by a regular expression

Claim 1. If a language is described by a regular expression, then it is regular

- Given a regular expression R, we show that there exists a finite automaton M with $L(M) = L(R)$

- We distinguish the following cases:

  1. $R = a$, $a \in \Sigma$

  2. $R = \epsilon$

  3. $R = \emptyset$

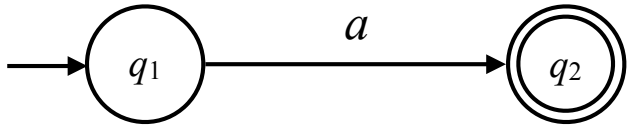
  4. $R = (R_1 \cup R_2)$, where R_1, R_2 are regular expressions

  5. $R = (R_1 R_2)$, where R_1, R_2 are regular expressions

  6. $R = (R_1^*)$ where R_1 is a regular expression

Case 1: $R = a$, $a \in \Sigma$
Show: $L(a)$ is regular

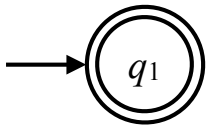
- $L(a) = \{a\}$
- NFA $N = (\{q_1, q_2\}, \Sigma, \delta, q_1, \{q_2\})$ with



  - $\delta(q_1, a) = \{q_2\}$
  - $\delta(r, b) = \varnothing$ for $(r, b) \neq (q_1, a)$

Case 2: $R = \epsilon$
Show: $L(R)$ is regular

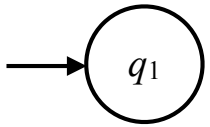
- $L(\epsilon) = \{\epsilon\}$
- NFA $N = (\{q_1\}, \Sigma, \delta, q_1, \{q_1\})$ with
  - $\delta(q_1, b) = \emptyset$ for any $b \in \Sigma \cup \{\epsilon\}$



Case 3: $R = \emptyset$

Show: $L(R)$ is regular

- $L(\emptyset) = \emptyset$
- NFA $N = (\{q_1\}, \Sigma, \delta, q_1, \emptyset)$ with



  - $\delta(q_1, b) = \emptyset$ for any $b \in \Sigma \cup \{\epsilon\}$

Case 4: $R = (R_1 \cup R_2)$, R_1, R_2 regular expressions; show: $L(R)$ is regular

- To show: Given regular expressions R_1 and R_2 where $L(R_1)$ and $L(R_2)$ are regular languages, also $L(R_1 \cup R_2)$ is regular
- We know: set of regular languages closed under union
  - Therefore: $L(R_1) \cup L\ (R_2)$ is regular
- From definition of languages recognized by regular expressions: $L(R_1 \cup R_2) = L(R_1) \cup L\ (R_2)$.
- Therefore, $L(R_1 \cup R_2)$ is regular

Case 5: $R = (R_1 R_2)$, R_1, R_2 regular expressions; show: $L(R)$ is regular

- To show: Given regular expressions R_1 and R_2 where $L(R_1)$ and $L(R_2)$ are regular languages, also $L(R_1 R_2)$ is regular
- We know: set of regular languages closed under concatenation
- Therefore $L(R_1)\ L\ (R_2)$ is regular
- From definition of languages recognized by regular expressions: $L(R_1 R_2) = L(R_1)\ L(R_2)$
- Therefore, $L(R_1 R_2)$ is regular

Case 6: $R = (R_1^*)$ where R_1 is a regular expression, show: $L(R)$ is regular

- To show: Given regular expression R_1 where $L(R_1)$ is a regular language, also $L(R_1)^*$ is regular
- We know: the set of regular languages closed under star
- Therefore: $L(R_1)^*$ is regular
- From definition of languages recognized by regular expressions: $L(R_1)^* = L(R_1^*)$
- Thus: language $L(R_1)^*$ described by regular expression R_1^* is regular

Equivalence of regular languages and the set of languages of regular expressions

Theorem: A language is regular if and only if there exists some regular expression that describes it

Proof consists of two parts

1. If a language is described by a regular expression, then it is regular

2. If a language is regular, then it can be described by a regular expression