CSE 311 Section 06

Models of Computation, Irregularity, Countability

Administrivia

Announcements & Reminders

- Today is the last section :(
  - Check your section attendance on canvas
- HW6 Part 1 due today 5/28 @ **6:00 PM**
- HW6 Part 2 due Monday 6/1 @ **6:00 PM**
- Quiz 6 on 6/4
- Final Exam
  - Monday 5/8 from **12:30-2:20 pm** in BAG 131 (Lecture A) and BAG 154 (Lecture B)
  - Bring your Husky ID

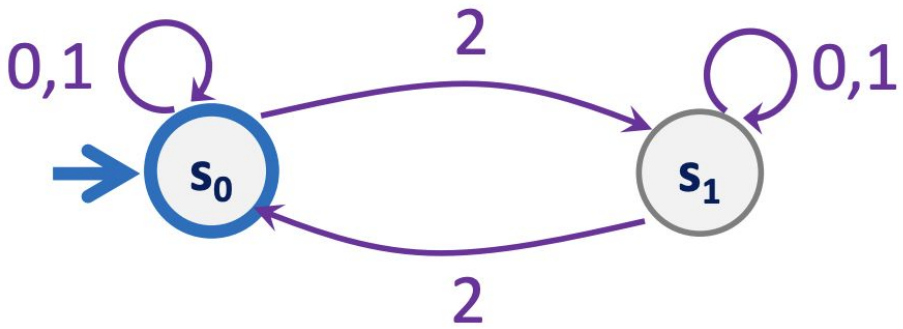
- **Course Evaluations will come out soon!**
  - Please consider taking 10 minutes to complete both section and course evaluations!

Deterministic Finite Automata

Deterministic Finite Automata

- A DFA is a finite-state machine that accepts or rejects a given string of symbols, by running through a state sequence uniquely determined by the string.

Strings with an even number of 2's

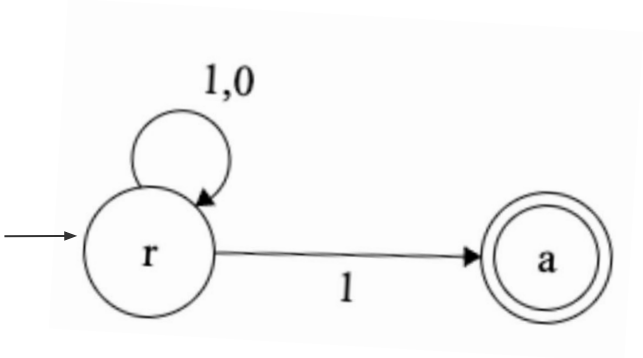


- An edge for every symbol in the language
- Every reject and accept state is handled

Nondeterministic Finite Automata

Nondeterministic Finite Automata

- **NFA in a nutshell** – A finite-state machine that explores all possibilities in parallel, it “knows” which path to take to get to an accept state. Specifies accept paths



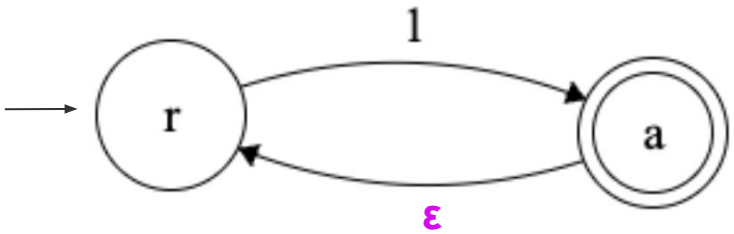
given two “1” edges, this will accept on 10101**1**

Nondeterministic Finite Automata

- **NFA in a nutshell** – A finite-state machine that explores all possibilities in parallel, it "knows" which path to take to get to an accept state. Specifies accept paths

- **ε-moves** – Special "free" hops that consume no input, letting the machine reposition before reading more symbols.

  think of strings with implicit **ε:**
  "11" as 1**ε1**

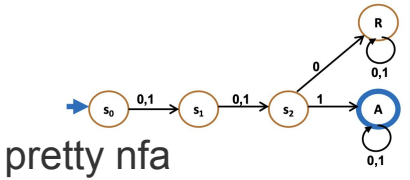


Nondeterministic Finite Automata

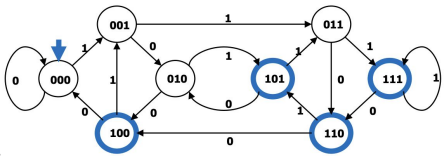
- **NFA in a nutshell** – A finite-state machine that explores all possibilities in parallel, it "knows" which path to take to get to an accept state. Specifies accept paths

- **ε-moves** – Special "free" hops that consume no input, letting the machine reposition before reading more symbols.

- **Key difference from a DFA** –
  - An NFA may have many (or zero) edges with the same label from one state, plus ε-edges
  - DFA has **exactly one per symbol** and **no ε-edges**
  - Makes for nicer looking machines!



pretty nfa



ugly dfa

Regular Expressions

Regular Expressions

ε matches only the **empty string**

a matches only the one-character string *a*

A ∪ **B** matches all strings that either **A** matches or **B** matches (or both)

AB matches all strings that have a first part that **A** matches followed by a second part that **B** matches

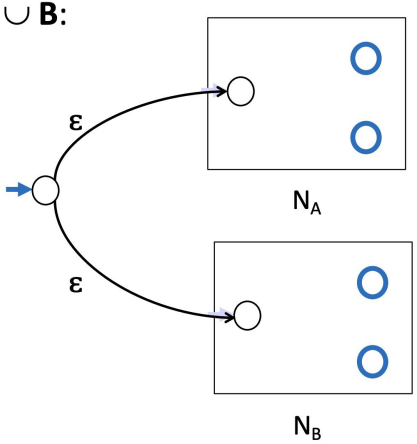
A* matches all strings that have any number of strings (even 0) that **A** matches, one after another (ε ∪ **A** ∪ **AA** ∪ **AAA** ∪ **…**)

Definition of the *language* matched by a regular expression

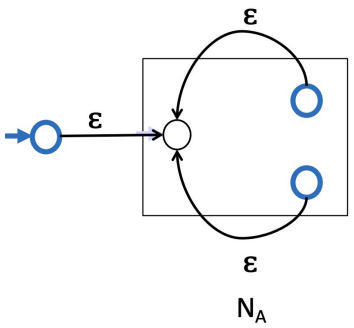
Regex to NFA

Regex to NFA Conversion!

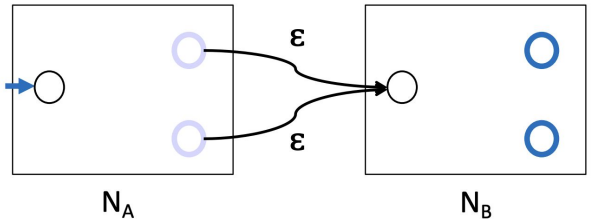
Case A ∪ B:



Case A*



Case AB:



Task 1: RE to NFA

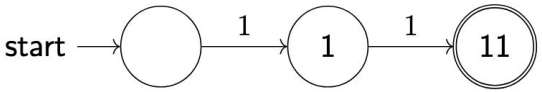
a) Convert the regular expression "$(11 \cup (01)^*)00$" to an NFA using the algorithm from lecture. You may skip adding ε-transitions for concatenation if they are obviously unnecessary, but otherwise, you should *precisely* follow the construction from lecture.

Work on this problem with the people around you

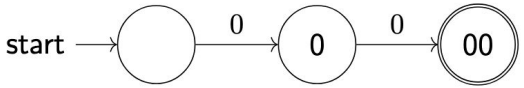
Task 1: RE to NFA

a) Convert the regular expression "$(11 \cup (01)^*)00$" to an NFA using the algorithm from lecture. You may skip adding ε-transitions for concatenation if they are obviously unnecessary, but otherwise, you should *precisely* follow the construction from lecture.

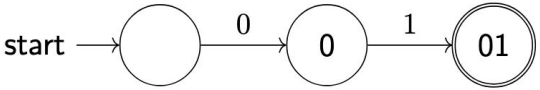
NFA for 11



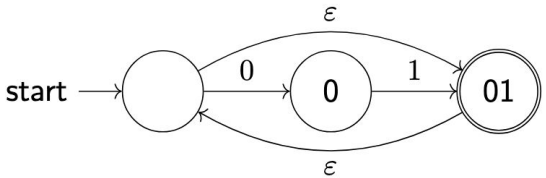
NFA for 00



NFA for 01

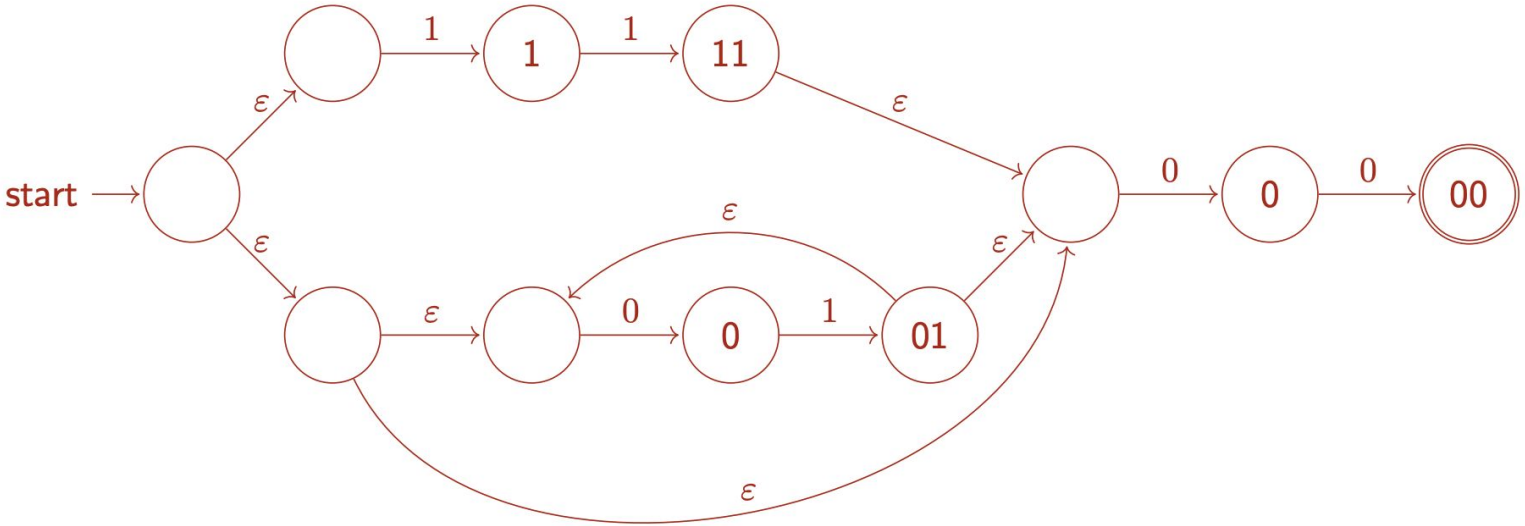


NFA for (01)*



Task 1: RE to NFA

a) Convert the regular expression "$(11 \cup (01)^*)00$" to an NFA using the algorithm from lecture. You may skip adding ε-transitions for concatenation if they are obviously unnecessary, but otherwise, you should *precisely* follow the construction from lecture.



Irregularity

Irregularity Template

Claim: *L* is an irregular language.

Proof: Suppose, for the sake of contradiction, that *L* is regular. Then there is a DFA M such that M accepts exactly *L*.

Consider S = [TODO] (S is an infinite set of strings)
Because the DFA is finite and S is infinite, there are two (different) strings x, y in S such that x and y go to the same state when read by M. [TODO] (We don't get to choose x, y, but we can describe them based on that set S we just defined)

Consider the string z = [TODO] (We do get to choose z depending on x, y)

Since x, y led to the same state and M is deterministic, xz and yz will also lead to the same state q in M. Observe that xz = [TODO], so $xz \in L$ but yz = [TODO], so $yz \notin L$. Since q can only be either an accept or a reject state, but not both, M does not actually recognize *L*. That's a contradiction!

Therefore, *L* is an irregular language.

Task 2a

Claim: $L = \{0^n1^n0^n : n \geq 0\}$ is an irregular language.

Proof: Suppose, for the sake of contradiction, that $L = \{0^n1^n0^n : n \geq 0\}$ is regular. Then there is a DFA M such that M accepts exactly L.

Consider S = [TODO] (S is an infinite set of strings)
Because the DFA is finite and S is infinite, there are two (different) strings x, y in S such that x and y go to the same state when read by M. [TODO] (We don't get to choose x, y, but we can describe them based on that set S we just defined)

Consider the string z = [TODO] (We do get to choose z depending on x, y)

Since x, y led to the same state and M is deterministic, xz and yz will also lead to the same state q in M. Observe that xz = [TODO], so $xz \in L$ but yz = [TODO], so $yz \notin L$. Since q can only be either an accept or a reject state, but not both, M does not actually recognize L. That's a contradiction!

Therefore, L is an irregular language.

Fill out this template for part a!

Task 2a

Claim: $L = \{0^n1^n0^n : n \geq 0\}$ is an irregular language.

Proof: Suppose, for the sake of contradiction, that $L = \{0^n1^n0^n : n \geq 0\}$ is regular. Then there is a DFA M such that M accepts exactly L.

Consider $S = \{0^n1^n : n \geq 0\}$
Because the DFA is finite and S is infinite, there are two (different) strings x, y in S such that x and y go to the same state when read by M. [TODO] (We don't get to choose x, y, but we can describe them based on that set S we just defined)

Consider the string z = [TODO] (We do get to choose z depending on x, y)

Since x, y led to the same state and M is deterministic, xz and yz will also lead to the same state q in M. Observe that xz = [TODO], so $xz \in L$ but yz = [TODO], so $yz \notin L$. Since q can only be either an accept or a reject state, but not both, M does not actually recognize L. That's a contradiction!

Therefore, L is an irregular language.

Task 2a

Claim: $L = \{0^n1^n0^n : n \geq 0\}$ is an irregular language.

Proof: Suppose, for the sake of contradiction, that $L = \{0^n1^n0^n : n \geq 0\}$ is regular. Then there is a DFA M such that M accepts exactly L.

Consider $S = \{0^n1^n : n \geq 0\}$
Because the DFA is finite and S is infinite, there are two (different) strings x, y in S such that x and y go to the same state when read by M. Since both are in S, $x = 0^a1^a$ for some integer $a \geq 0$, and $y = 0^b1^b$ for some integer $b \geq 0$, with $a \neq b$.

Consider the string z = [TODO] (We do get to choose z depending on x, y)

Since x, y led to the same state and M is deterministic, xz and yz will also lead to the same state q in M. Observe that xz = [TODO], so $xz \in L$ but yz = [TODO], so $yz \notin L$. Since q can only be either an accept or a reject state, but not both, M does not actually recognize L. That's a contradiction!

Therefore, L is an irregular language.

Task 2a

Claim: $L = \{0^n1^n0^n : n \geq 0\}$ is an irregular language.

Proof: Suppose, for the sake of contradiction, that $L = \{0^n1^n0^n : n \geq 0\}$ is regular. Then there is a DFA M such that M accepts exactly L.

Consider $S = \{0^n1^n : n \geq 0\}$
Because the DFA is finite and S is infinite, there are two (different) strings x, y in S such that x and y go to the same state when read by M. Since both are in S, $x = 0^a1^a$ for some integer $a \geq 0$, and $y = 0^b1^b$ for some integer $b \geq 0$, with $a \neq b$.

Consider the string $z = 0^a$

Since x, y led to the same state and M is deterministic, xz and yz will also lead to the same state q in M. Observe that $xz = 0^a1^a0^a$, so $xz \in L$ but $yz = 0^b1^b0^a$, so $yz \notin L$. Since q can only be either an accept or a reject state, but not both, M does not actually recognize L. That's a contradiction!

Therefore, L is an irregular language.

Irregularity Tips

- Many correct choices for the infinite set S of partial prefix strings.
  - S doesn't need to account for all partial strings; it can be a subset. It does need to be **infinite**.
- You **do not** get to choose which two prefix strings end up at the same intermediate state of the DFA.
  - But you do know **they are in S** and that they are **distinct**.
- You **do** get to choose the common suffix string to append based on the two prefix strings that ended up in the same state. Choose wisely, figure out what the DFA needs to "count" :D
- Template for irregularity
  - Choose S, common string to append (based on prefix string structure), argue why one string in the language and other isn't

Countability

Countability

Recall from lecture…

Definition: A set is **countable** iff it has the same cardinality as some subset of $\mathbb{N}$.

Equivalent: A set S is countable iff there is an *onto* function $g : \mathbb{N} \to S$

Equivalent: A set S is countable iff we can order the elements $S = \{x_1, x_2, x_3, \ldots\}$

Countable and Uncountable Sets Examples

Countable Sets:

- $\mathbb{N}$ - the set of Natural Numbers; $\mathbb{N}$ = {0, 1, 2, …}
- $\mathbb{Z}$ - the set of Integers; $\mathbb{Z}$ = {…, -2, -1, 0, 1, 2, …}
- $\mathbb{Q}$ - the set of Rational Numbers; e.g. ½, -17, 32/48
- [n] - the set {1, 2, …, n} where n is a natural number

Uncountable Sets:

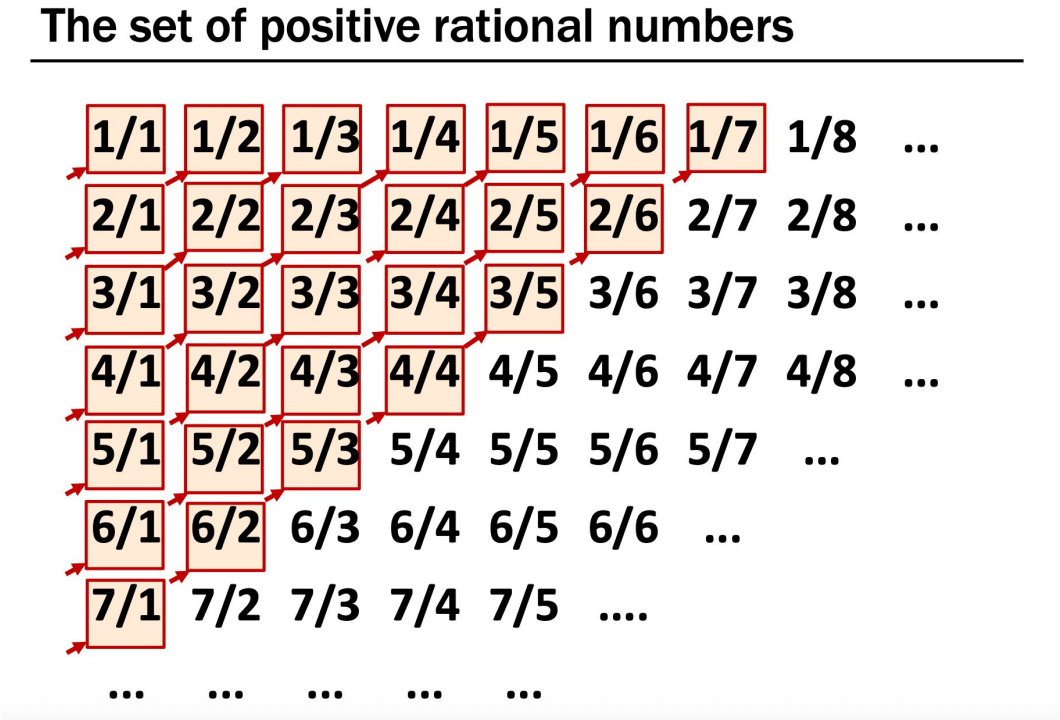
- $\mathbb{R}$ - the set of Real Numbers; e.g. 1, -17, 32/48, π, e, √2
- $\mathbb{R} \setminus \mathbb{Q}$ - the set of Irrational Numbers; e.g. π, e, √2

How to Prove Countability

- Enumeration
  - Show how to list (enumerate) the elements of S
- Dovetailing
  - Explained on the next slide
- Subset
  - The set S is countable because it is a subset of a countable set
- One-to-one/onto
  - Construct a surjection g (onto function) from a countable set X (that is, one with an existing surjection h from $\mathbb{N}$ to itself) to the set S you're trying to prove is countable
  - Since there is a surjection f from $\mathbb{N}$ to S (namely, $f = g \circ h$), S is countable
  - Alternatively, construct an injection (one-to-one function) from S to the countable set X

Dovetailing

- Put all elements into finite groups
- Order the groups
- List elements in order by group (arbitrary order within each group)
- In the example to the right, each diagonal is a finite group



Countability Template (Enumeration)

Consider the following enumeration of elements of S. [TODO: construct the enumeration, commonly done using dovetailing]

Thus, since we can enumerate the elements of S, it must be countable.

Countability Template (Subset)

Consider an arbitrary element $y \in S$. [TODO: show how y is an element of a known countable set X, such as $\mathbb{N}$]

Since y was arbitrary, by definition of subset, S is a subset of the countable set X.

Thus, S must be countable, because it is a subset of a countable set.

Countability Template (Surjection)

Consider the function f: $\mathbb{N} \rightarrow$ S. [TODO: Define how f was constructed]

Let y be an arbitrary element of S. Because of the way f was constructed [TODO: explain why], there exists a natural number x such that f(x)=y.

Since y was arbitrary, the function f is onto (aka a surjection).

So, there exists a surjection f that maps natural numbers to elements of S. Thus, S is countable.

A function $f : A \rightarrow B$ is onto if for every $y \in B$, there exists $x \in A$ such that $f(x) = y$.

Countability Template (Injection)

Consider the function f: S → ℕ. [TODO: Define how f was constructed]

Let x_1, x_2 be arbitrary elements of S. Suppose $f(x_1) = f(x_2)$. Because of the way f was constructed [TODO: explain why], this must mean $x_1 = x_2$.

Since x_1, x_2 were arbitrary, the function f is one-to-one (aka an injection).

So, there exists an injection f that maps elements of S to natural numbers. Thus, S is countable.

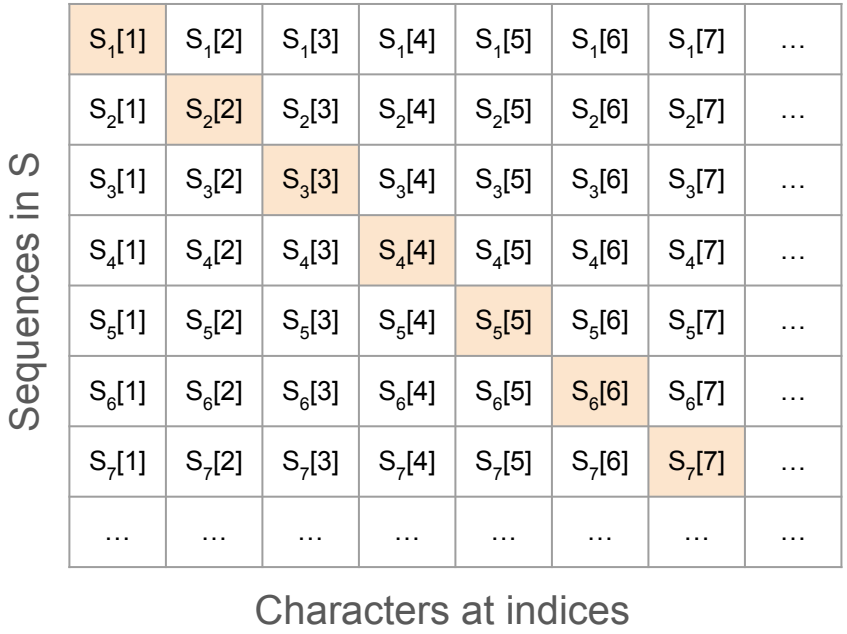
A function $f : A \to B$ is one-to-one if for every x_1, $x_2 \in A$, if $f(x_1) = f(x_2) = y$, then $x_1 = x_2$.

How to Prove Uncountability: Diagonalization

- Misleadingly similar to dovetailing (diagonalization has the opposite goal)
- Construct an element x of set S that would not be accounted for by any enumeration strategy nor any surjective function
- For every element S_i of set S that was counted by an enumeration strategy, make the element x be such that it couldn't possibly be the counted S_i

Diagonalization

- Each element in S is an **infinite** sequence or can be represented as an **infinite** sequence
  - If all of the elements are finite sequences or can be represented by finite sequences, then this technique won't work and the set S is (most likely) countable
- The rows are the enumerated sequences S_i
- The rows consist of the elements in S_i. The ith row in the jth column is the character located at the jth position of the sequence S_i.
- Diagonalization: Construct x so that for each element S_i, it doesn't match on the ith character.



$S_1[1]$	$S_1[2]$	$S_1[3]$	$S_1[4]$	$S_1[5]$	$S_1[6]$	$S_1[7]$	…
$S_2[1]$	$S_2[2]$	$S_2[3]$	$S_2[4]$	$S_2[5]$	$S_2[6]$	$S_2[7]$	…
$S_3[1]$	$S_3[2]$	$S_3[3]$	$S_3[4]$	$S_3[5]$	$S_3[6]$	$S_3[7]$	…
$S_4[1]$	$S_4[2]$	$S_4[3]$	$S_4[4]$	$S_4[5]$	$S_4[6]$	$S_4[7]$	…
$S_5[1]$	$S_5[2]$	$S_5[3]$	$S_5[4]$	$S_5[5]$	$S_5[6]$	$S_5[7]$	…
$S_6[1]$	$S_6[2]$	$S_6[3]$	$S_6[4]$	$S_6[5]$	$S_6[6]$	$S_6[7]$	…
$S_7[1]$	$S_7[2]$	$S_7[3]$	$S_7[4]$	$S_7[5]$	$S_7[6]$	$S_7[7]$	…
…	…	…	…	…	…	…	…

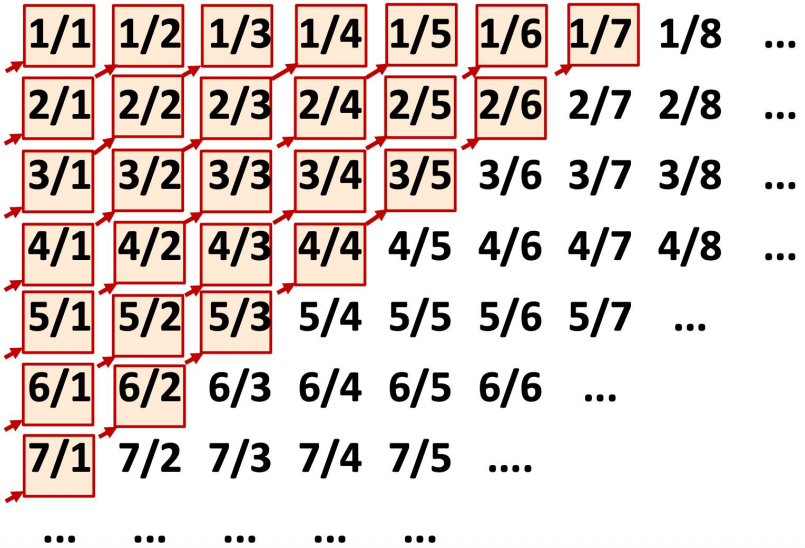
Dovetailing vs Diagonalization

Diagonalization: To prove **un**countability

$S_1[1]$	$S_1[2]$	$S_1[3]$	$S_1[4]$	$S_1[5]$	$S_1[6]$	$S_1[7]$	…
$S_2[1]$	$S_2[2]$	$S_2[3]$	$S_2[4]$	$S_2[5]$	$S_2[6]$	$S_2[7]$	…
$S_3[1]$	$S_3[2]$	$S_3[3]$	$S_3[4]$	$S_3[5]$	$S_3[6]$	$S_3[7]$	…
$S_4[1]$	$S_4[2]$	$S_4[3]$	$S_4[4]$	$S_4[5]$	$S_4[6]$	$S_4[7]$	…
$S_5[1]$	$S_5[2]$	$S_5[3]$	$S_5[4]$	$S_5[5]$	$S_5[6]$	$S_5[7]$	…
$S_6[1]$	$S_6[2]$	$S_6[3]$	$S_6[4]$	$S_6[5]$	$S_6[6]$	$S_6[7]$	…
$S_7[1]$	$S_7[2]$	$S_7[3]$	$S_7[4]$	$S_7[5]$	$S_7[6]$	$S_7[7]$	…
…	…	…	…	…	…	…	…

Dovetailing: To prove countability

The set of positive rational numbers



Uncountability Template

Suppose, for the sake of contradiction, that S is countable.

Then, there exists a surjection (onto function) f: $\mathbb{N}$ -> S (which means we can enumerate elements S_i in S). So for every natural number i, we have some element S_i that i maps to.

We now construct an element x of S (in such a way such that no i maps to x) [TODO: construct the element x (commonly done via diagonalization)].

So, no f(i) maps to the element x of S. But then f is not a surjection and there exists no way for us to enumerate all elements of S. That's a contradiction! Therefore, S is uncountable.

Task 3

b) Prove that $P(\mathbb{N})$ is uncountable.

c) Show that $\mathbb{N} \times \mathbb{N}$ is countable.

Work on this problem with the people around you

Task 3

b) Prove that $P(\mathbb{N})$ is uncountable.

Task 3

b) Prove that $P(\mathbb{N})$ is uncountable.

Suppose, for the sake of contradiction, that $P(\mathbb{N})$ is countable.

Task 3

b) Prove that $P(\mathbb{N})$ is uncountable.

Suppose, for the sake of contradiction, that $P(\mathbb{N})$ is countable.

Then, there exists a surjection (onto function) $f: \mathbb{N} \to S$. This means we can define an enumeration of elements S_i in $P(\mathbb{N})$.

Task 3

b) Prove that P(ℕ) is uncountable.

Suppose, for the sake of contradiction, that P(ℕ) is countable.

Then, there exists a surjection (onto function) f: ℕ -> S. This means we can define an enumeration of elements S_i in P(ℕ).

Let s_i be the binary set representation of S_i in P(ℕ) (each 1 or 0 represents the inclusion/exclusion of each element of ℕ in S_i). For example, for the set {0,1,2}, the binary set representation would be 111000…

We then construct a new subset X ⊆ ℕ such that x[i] = ~s_i[i] (that is, x[i] is 1 if s_i[i] is 0, and x[i] is 0 otherwise).

For example,
s_1 = 001…, then x = 1…
s_2 = 010…, then x = 10…
s_3 = 110…, then x = 101…
Etc…

Task 3

b) Prove that $P(\mathbb{N})$ is uncountable.

Suppose, for the sake of contradiction, that $P(\mathbb{N})$ is countable.

Then, there exists a surjection (onto function) $f: \mathbb{N} \rightarrow S$. This means we can define an enumeration of elements S_i in $P(\mathbb{N})$.

Let s_i be the binary set representation of S_i in $P(\mathbb{N})$ (each 1 or 0 represents the inclusion/exclusion of each element of $\mathbb{N}$ in S_i). For example, for the set {0,1,2}, the binary set representation would be 111000…

We then construct a new subset $X \subseteq \mathbb{N}$ such that $x[i] = \sim s_i[i]$ (that is, $x[i]$ is 1 if $s_i[i]$ is 0, and $x[i]$ is 0 otherwise).

Note that X is not any of S_i, since it differs from S_i on the *i*th natural number. However, X still represents a valid subset of the natural numbers.

Task 3

b) Prove that P(ℕ) is uncountable.

Suppose, for the sake of contradiction, that P(ℕ) is countable.

Then, there exists a surjection (onto function) f: ℕ -> S. This means we can define an enumeration of elements S_i in P(ℕ).

Let s_i be the binary set representation of S_i in P(ℕ) (each 1 or 0 represents the inclusion/exclusion of each element of ℕ in S_i). For example, for the set {0,1,2}, the binary set representation would be 111000…

We then construct a new subset X ⊆ ℕ such that x[i] = ~s_i[i] (that is, x[i] is 1 if s_i[i] is 0, and x[i] is 0 otherwise).

Note that X is not any of S_i, since it differs from S_i on the *i*th natural number. However, X still represents a valid subset of the natural numbers.

So, no f(i)=S_i maps to the element X in P(ℕ). But then f is not a surjection and there exists no way for us to enumerate all elements of P(ℕ). That's a contradiction! Therefore, P(ℕ) is uncountable.

Task 3

c) Show that $\mathbb{N} \times \mathbb{N}$ is countable.

We use dovetailing to create a sequence of elements of $\mathbb{N} \times \mathbb{N}$ that includes the entirety of $\mathbb{N} \times \mathbb{N}$.

For a fixed integer $k \geq 2$, consider subset S_k of $\mathbb{N} \times \mathbb{N}$ consisting of the elements (a, b) such that $a + b = k$. There can be at most k - 1 such elements because for each value of $a = 1, 2, \ldots, k - 1$, there can only be one possible value for b, namely k - a. Thus, if we create a sequence consisting of all the elements of S_2, then S_3, then S_4, etc. because each set is of finite size, any pair $(a, b) \in \mathbb{N} \times \mathbb{N}$ will eventually show up in this sequence in S_{a+b}.

Thus, since we can enumerate the elements of $\mathbb{N} \times \mathbb{N}$, it must be countable.

(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)	…
(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)	…
(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)	…
(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	(4,6)	(4,7)	…
(5,1)	(5,2)	(5,3)	(5,4)	(5,5)	(5,6)	(5,7)	…
(6,1)	(6,2)	(6,3)	(6,4)	(6,5)	(6,6)	(6,7)	…
(7,1)	(7,2)	(7,3)	(7,4)	(7,5)	(7,6)	(7,7)	…
…	…	…	…	…	…	…	…

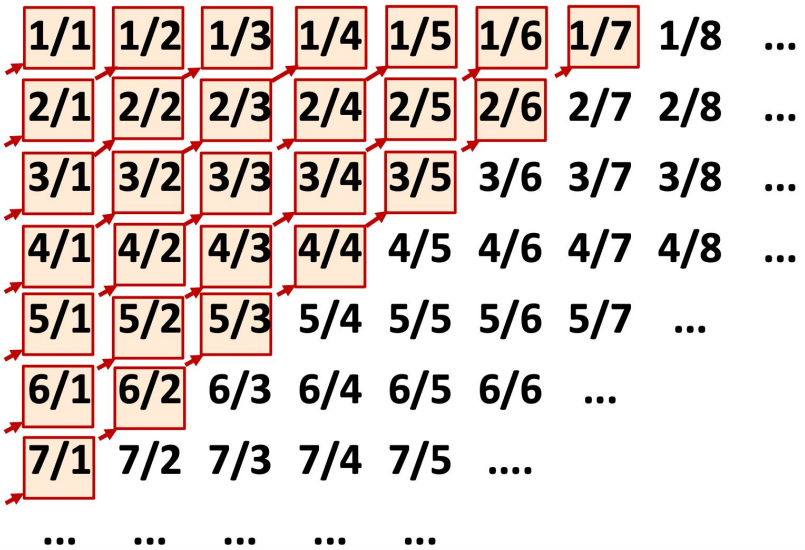
Task 3

Looks familiar?

The set $\mathbb{N} \times \mathbb{N}$

(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)	…
(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)	…
(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)	…
(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	(4,6)	(4,7)	…
(5,1)	(5,2)	(5,3)	(5,4)	(5,5)	(5,6)	(5,7)	…
(6,1)	(6,2)	(6,3)	(6,4)	(6,5)	(6,6)	(6,7)	…
(7,1)	(7,2)	(7,3)	(7,4)	(7,5)	(7,6)	(7,7)	…
…	…	…	…	…	…	…	…

The set of positive rational numbers



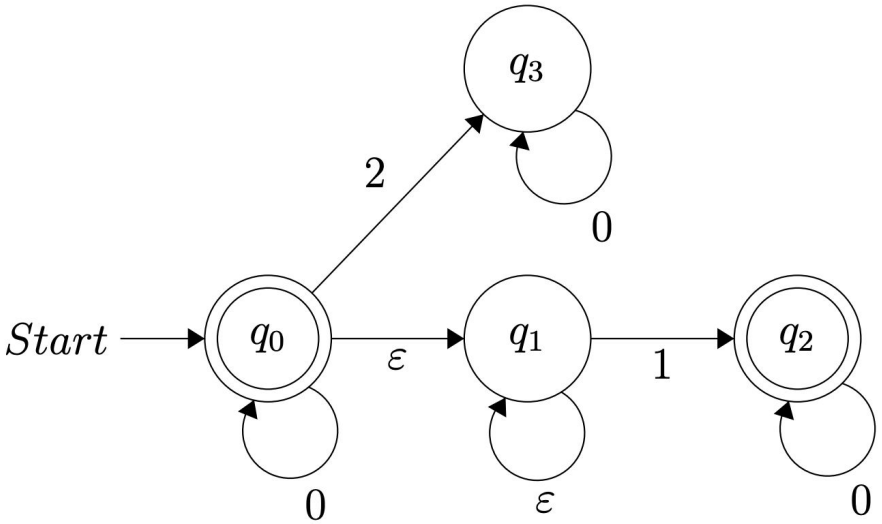
NFA to DFA

DFAs and NFAs

- Any language that can be accepted by a DFA can be accepted by an NFA
- A DFA is an NFA!
- Every language that can be accepted by an NFA can be accepted by a DFA also!
- Using the following algorithm:
  - Start at the set of states which can be reached from the empty state
  - Construct a table with all symbols of the ∑, and what each symbol takes the set of states to
  - Keep going until there are no more unique states

Task 4

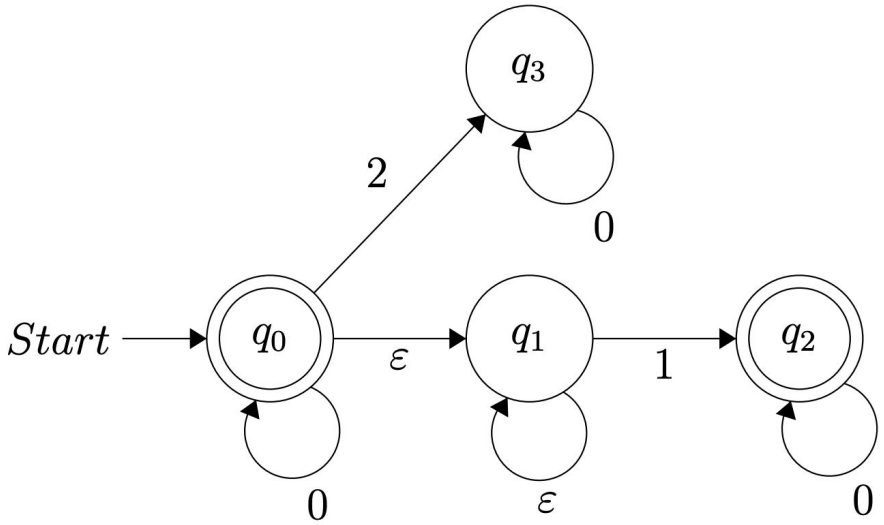
Convert the following NFA to a DFA of the same language



Task 4 - Documenting states in a table

Where can we get from the start state (q0) without reading input?

(DFA) state	0-transition	1-transition	2-transition

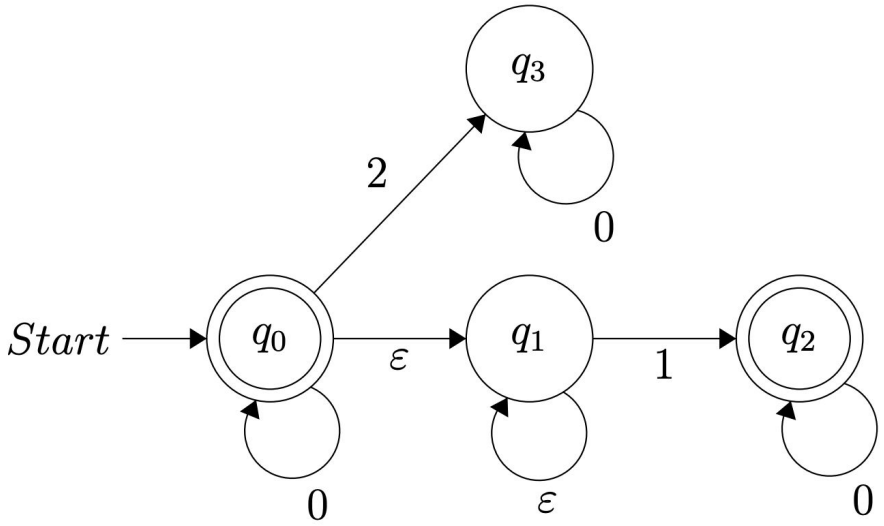


Task 4 - Documenting states in a table

Where can we get from the start state (q0) without reading input?

- q0
- take ε to q1

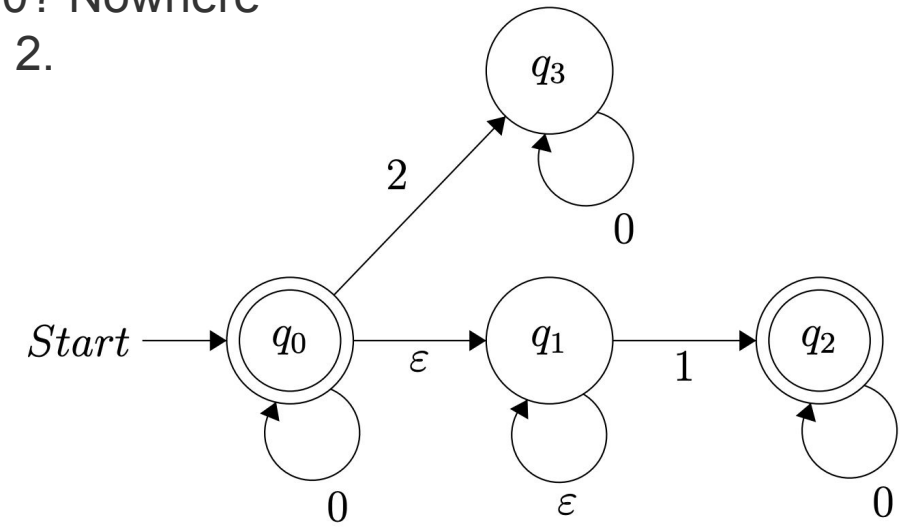
(DFA) state	0-transition	1-transition	2-transition
q0, q1			



Task 4 - Documenting states in a table

From q0, where can we get reading in 0? q0 and q1
From q1, where can we get reading in 0? Nowhere
Repeat for reading in 1, and reading in 2.
Then fill in the state table.

(DFA) state	0-transition	1-transition	2-transition
q0, q1	q0, q1	q2	q3

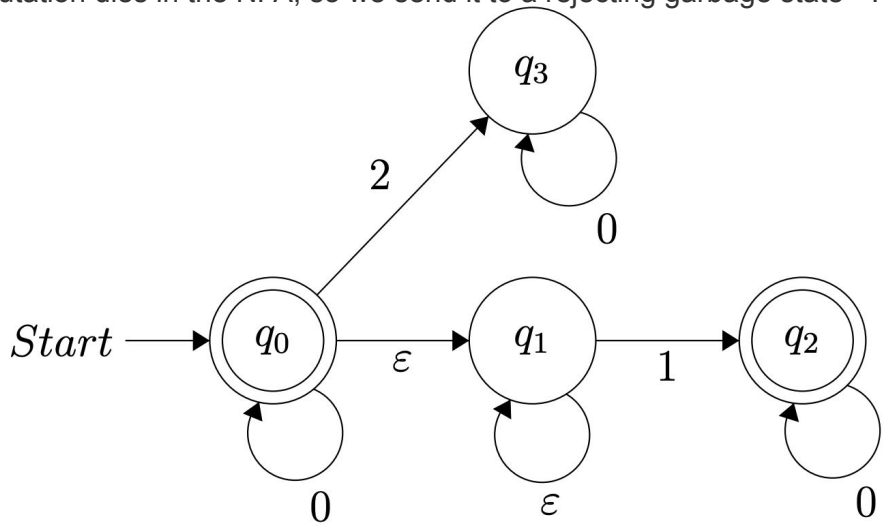


Task 4 - Documenting states in a table

We need to define the other states (q2, q3) in the table now.

- When there's no defined transition for an input, the computation dies in the NFA, so we send it to a rejecting garbage state ∅.

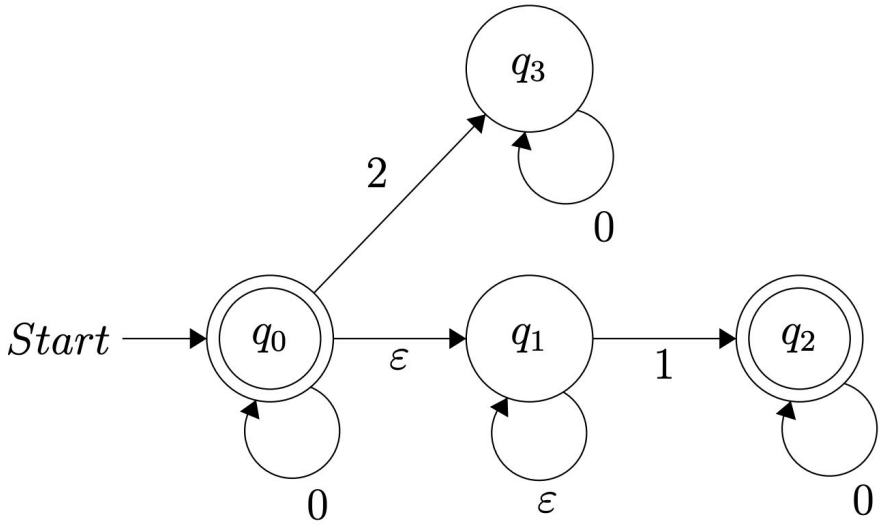
(DFA) state	0-transition	1-transition	2-transition
q0, q1	q0, q1	q2	q3
q2	q2	∅	∅
q3			



Task 4 - Documenting states in a table

We need to define the other states (q2 and q3) in the table now.

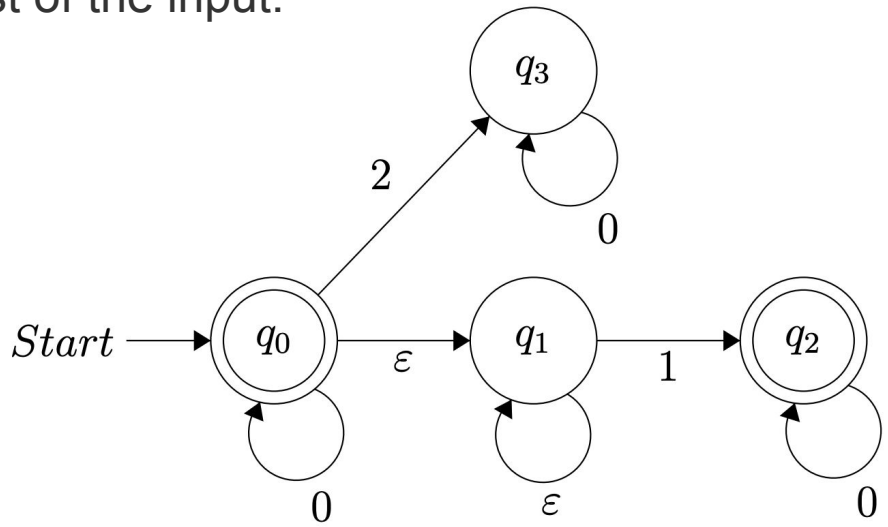
(DFA) state	0-transition	1-transition	2-transition
q0, q1	q0, q1	q2	q3
q2	q2	∅	∅
q3	q3	∅	∅



Task 4 - Documenting states in a table

The only state we have left to define is the “garbage” state. This one always is rejecting regardless of the rest of the input.

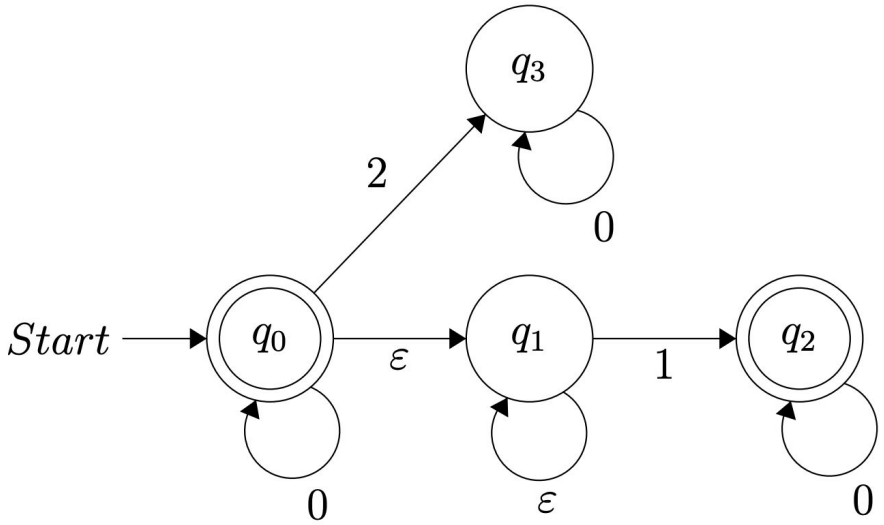
(DFA) state	0-transition	1-transition	2-transition
q0, q1	q0, q1	q2	q3
q2	q2	∅	∅
q3	q3	∅	∅
∅	∅	∅	∅



Task 4 - Documenting states in a table

Which states are accepting? (The rest are rejecting)

(DFA) state	0-transition	1-transition	2-transition
q0, q1	q0, q1	q2	q3
q2	q2	∅	∅
q3	q3	∅	∅
∅	∅	∅	∅

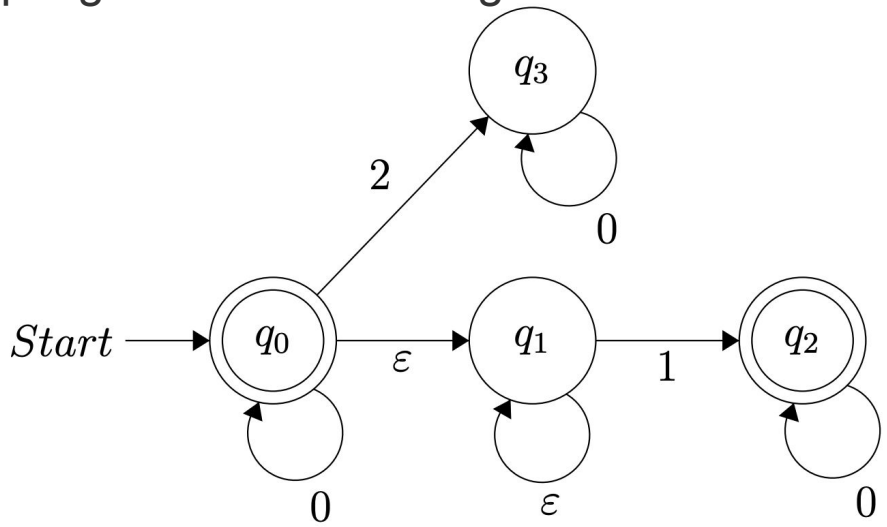


Task 4 - Documenting states in a table

Which states are accepting? (The rest are rejecting)

- All DFA states that include an accepting state from the original NFA.

(DFA) state	0-transition	1-transition	2-transition
q0, q1	q0, q1	q2	q3
q2	q2	∅	∅
q3	q3	∅	∅
∅	∅	∅	∅

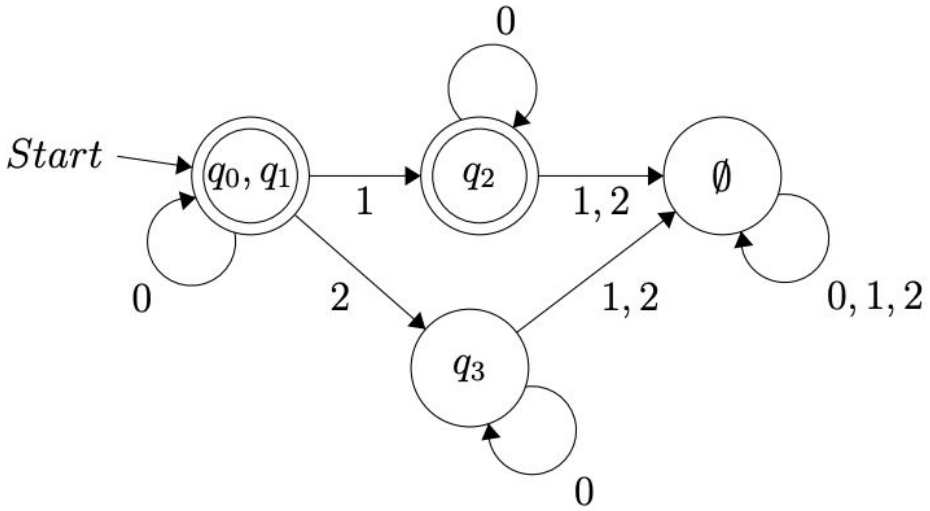


Now try drawing out the DFA using the state table.

- Don’t forget a start state!

Task 4

(DFA) state	0-transition	1-transition	2-transition
q0, q1	q0, q1	q2	q3
q2	q2	∅	∅
q3	q3	∅	∅
∅	∅	∅	∅



That's All, Folks!

Thanks for coming to section this week!
Any questions?