IN2080

2. Nondeterministic Automata

Einar Broch Johnsen

University of Oslo

1st February 2023



Department of Informatics



University of Oslo

Last Lecture

- **Basics:** sets, de Morgan's laws, tuples, products, functions, strings
- **Finite automata:**
  - Accepts or rejects strings (so-called *words*)
  - States, transitions, start state, accept states
  - Formal definitions of DFA, acceptance, regular language
  - Operations on languages: union, intersection, concatenation, Kleene star
  - Theorem: RL closed under union and intersection

Nondeterministic Finite Automata

For DFA, the transition function δ gave,
for a given state and input symbol,
precisely one following state.

$\rightarrow$ **determinism**

Nondeterministic Finite Automata

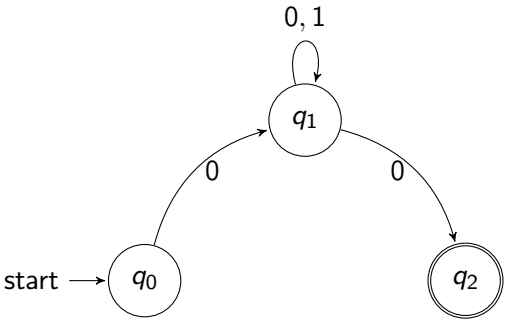
For DFA, the transition function δ gave,
for a given state and input symbol,
precisely one following state.

$\rightarrow$ **determinism**

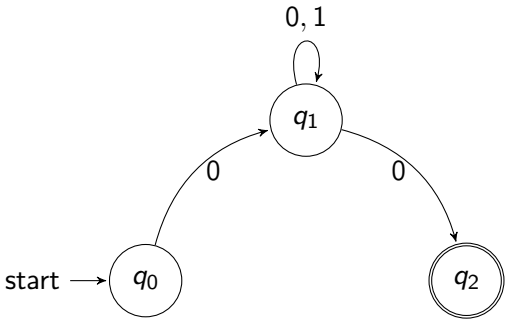
Now we allow for multiple possible "next" states.

$\rightarrow$ **nondeterminism**

NFA - An Example

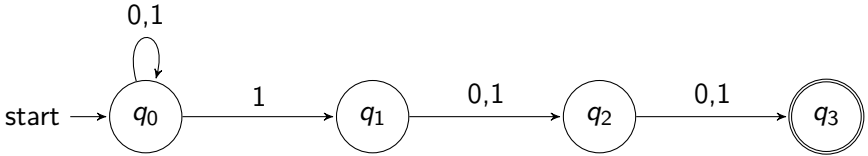


NFA - An Example

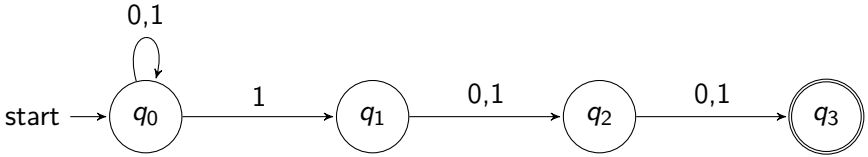


This language consists of all 0,1 sequences starting and ending with 0.

NFA - Another Example

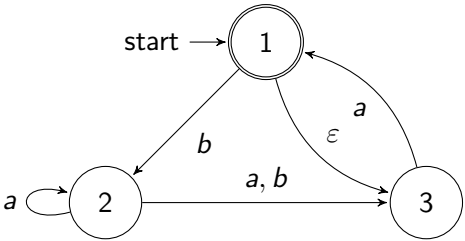


NFA - Another Example



This language consists of all 0,1 sequences with a 1 in the third position from the end

NFA - an Example with Empty Transitions



Nondeterministic Finite Automata

Definition

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

1. Q is a finite set of states,
2. Σ is a finite alphabet,
3. $\delta : Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$ is the transition function, and
4. $F \subseteq Q$ is the set of accepting states.

Acceptance for Nondeterministic Finite Automata

- What is the acceptance condition for NFA?

Acceptance for Nondeterministic Finite Automata

- What is the acceptance condition for NFA?
- Remember acceptance for DFA: a sequence of states such that the transition function gives us the next state in the sequence

Definition (DFA)

A DFA $(Q, \Sigma, \delta, q_0, F)$ *accepts* an input $w = w_1 w_2 \cdots w_n$ if there exists a sequence of states $s_0 \cdots s_n$ such that

1. s_0 is the start state q_0
2. $\delta(s_i, w_{i+1}) = s_{i+1}$ (a valid transition is chosen for the currently parsed input symbol)
3. $s_n \in F$, i.e., is an accept state.

Acceptance for Nondeterministic Finite Automata

- What is the acceptance condition for NFA?
- Remember acceptance for DFA: a sequence of states such that the transition function gives us the next state in the sequence
- For NFA, *one* of the next states in each transition must be the next state in the sequence

Definition (DFA)

A DFA $(Q, \Sigma, \delta, q_0, F)$ *accepts* an input $w = w_1 w_2 \cdots w_n$ if there exists a sequence of states $s_0 \cdots s_n$ such that

1. s_0 is the start state q_0
2. $\delta(s_i, w_{i+1}) = s_{i+1}$ (a valid transition is chosen for the currently parsed input symbol)
3. $s_n \in F$, i.e., is an accept state.

Acceptance for Nondeterministic Finite Automata

- What is the acceptance condition for NFA?
- Remember acceptance for DFA: a sequence of states such that the transition function gives us the next state in the sequence
- For NFA, *one* of the next states in each transition must be the next state in the sequence

Definition (NFA)

An NFA $(Q, \Sigma, \delta, q_0, F)$ *accepts* an input $w = w_1 w_2 \cdots w_n$ if there exists a sequence of states $s_0 \cdots s_n$ such that

1. s_0 is the start state q_0
2. $s_{i+1} \in \delta(s_i, w_{i+1})$ (a valid next state is chosen for the currently parsed input symbol)
3. $s_n \in F$, i.e., is an accept state.

NFA

- Notice that DFAs are special cases of NFAs.

NFA

- Notice that DFAs are special cases of NFAs.
- DFAs accept regular languages, but what languages do NFAs accept?

NFA

- Notice that DFAs are special cases of NFAs.
- DFAs accept regular languages, but what languages do NFAs accept?

- As it turns out: regular languages!

NFA

- Notice that DFAs are special cases of NFAs.
- DFAs accept regular languages, but what languages do NFAs accept?
- As it turns out: regular languages!
- In other words, in a sense, DFA=NFA.

DFA=NFA

Theorem

Every NFA $N = (Q, \Sigma, \delta, q_0, F)$ has an equivalent DFA $M = (Q', \Sigma, \delta', q_0', F')$.

DFA=NFA

Theorem

Every NFA $N = (Q, \Sigma, \delta, q_0, F)$ has an equivalent DFA $M = (Q', \Sigma, \delta', q_0', F')$.

Proof idea:

- Each state in the NFA can have multiple possible following states.
- We need to simultaneously keep track of all these possible following states in *one* state in the DFA.

DFA=NFA

Theorem

Every NFA $N = (Q, \Sigma, \delta, q_0, F)$ has an equivalent DFA $M = (Q', \Sigma, \delta', q_0', F')$.

Proof idea:

- Each state in the NFA can have multiple possible following states.
- We need to simultaneously keep track of all these possible following states in *one* state in the DFA.
- Since the "set of possible following states" in the NFA could be any subset of the state set Q, the DFA's state set Q' must be $\mathcal{P}(Q)$.

DFA=NFA

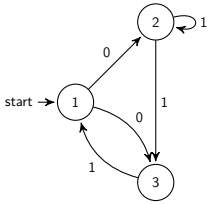
Theorem

Every NFA $N = (Q, \Sigma, \delta, q_0, F)$ has an equivalent DFA $M = (Q', \Sigma, \delta', q_0', F')$.

Proof idea:

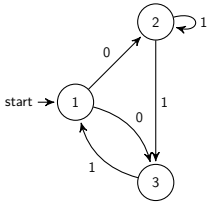
- Each state in the NFA can have multiple possible following states.
- We need to simultaneously keep track of all these possible following states in *one* state in the DFA.
- Since the "set of possible following states" in the NFA could be any subset of the state set Q, the DFA's state set Q' must be $\mathcal{P}(Q)$.
- Let us first assume there are no ε transitions.

DFA=NFA: Example



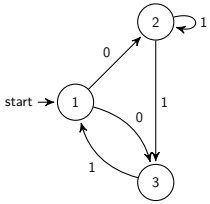
- Let us construct the DFA.

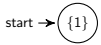
DFA=NFA: Example



- Let us construct the DFA.
- Clearly $q_0' = \{q_0\}$.

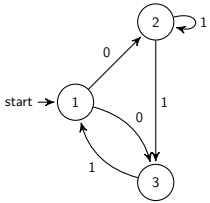
DFA=NFA: Example

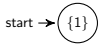




- Let us construct the DFA.
- Clearly $q_0' = \{q_0\}$.

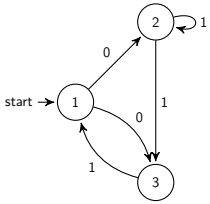
DFA=NFA: Example

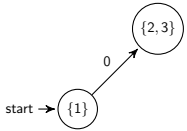




- Let us construct the DFA.
- Clearly $q_0' = \{q_0\}$.
- Are there any transitions from the state $\{1\}$?

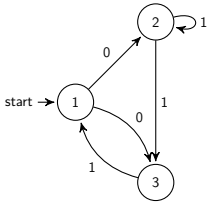
DFA=NFA: Example

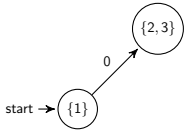




- Let us construct the DFA.
- Clearly $q_0' = \{q_0\}$.
- Are there any transitions from the state $\{1\}$?

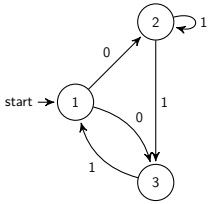
DFA=NFA: Example

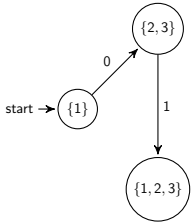




- Let us construct the DFA.
- Clearly $q_0' = \{q_0\}$.
- Are there any transitions from the state $\{1\}$?
- Are there any transitions from the state $\{2, 3\}$?

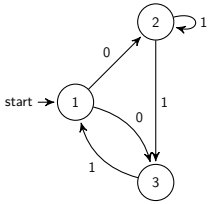
DFA=NFA: Example

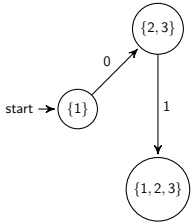




- Let us construct the DFA.
- Clearly $q_0' = \{q_0\}$.
- Are there any transitions from the state $\{1\}$?
- Are there any transitions from the state $\{2, 3\}$?

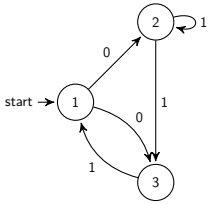
DFA=NFA: Example

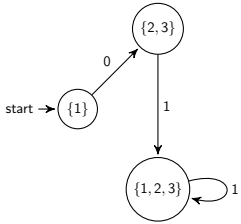




- Let us construct the DFA.
- Clearly $q_0' = \{q_0\}$.
- Are there any transitions from the state $\{1\}$?
- Are there any transitions from the state $\{2, 3\}$?
- Are there any transitions from the state $\{1, 2, 3\}$?

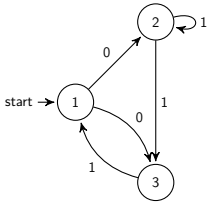
DFA=NFA: Example

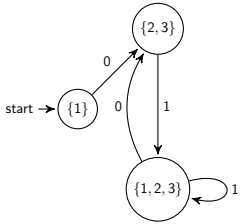




- Let us construct the DFA.
- Clearly $q_0' = \{q_0\}$.
- Are there any transitions from the state $\{1\}$?
- Are there any transitions from the state $\{2, 3\}$?
- Are there any transitions from the state $\{1, 2, 3\}$?

DFA=NFA: Example





- Let us construct the DFA.
- Clearly $q_0' = \{q_0\}$.
- Are there any transitions from the state $\{1\}$?
- Are there any transitions from the state $\{2, 3\}$?
- Are there any transitions from the state $\{1, 2, 3\}$?

NFA=DFA

What about the transition function δ'?

NFA=DFA

What about the transition function δ'?

- A state R in the DFA M corresponds to a set of states in the NFA N.
- For an input w_i at state R, we need to consider *all* possible following states from the set of possible states R.

NFA=DFA

What about the transition function δ'?

- A state R in the DFA M corresponds to a set of states in the NFA N.
- For an input w_i at state R, we need to consider *all* possible following states from the set of possible states R. Or, more formally,

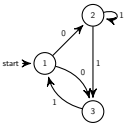
$$\begin{aligned}\delta'(R, w_i) &= \bigcup_{r \in R} \delta(r, w_i) \\ &= \{q \in Q \mid q \in \delta(r, w_i) \text{ for some } r \in R\}\end{aligned}$$

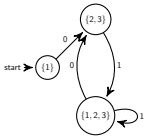
NFA=DFA

What about the transition function δ'?

- A state R in the DFA M corresponds to a set of states in the NFA N.
- For an input w_i at state R, we need to consider *all* possible following states from the set of possible states R. Or, more formally,

$$\begin{aligned}\delta'(R, w_i) &= \bigcup_{r \in R} \delta(r, w_i) \\ &= \{q \in Q \mid q \in \delta(r, w_i) \text{ for some } r \in R\}\end{aligned}$$





NFA=DFA

- Since an NFA accepts an input if *any* of the possible computations ends in an accept state, $F' = \{R \subseteq Q \mid R \text{ contains a state } r \in F\}$.

NFA=DFA

- Since an NFA accepts an input if *any* of the possible computations ends in an accept state, $F' = \{R \subseteq Q \mid R \text{ contains a state } r \in F\}$.

- **Almost done!** But we still need to take ε transitions into account. To that end, let

$$E(R) = \{q \mid q \text{ can be reached from } R \text{ with 0 or more } \varepsilon \text{ transitions}\} \text{ for } R \subseteq Q.$$

NFA=DFA

- Since an NFA accepts an input if *any* of the possible computations ends in an accept state, $F' = \{R \subseteq Q \mid R \text{ contains a state } r \in F\}$.
- **Almost done!** But we still need to take ε transitions into account. To that end, let

$$E(R) = \{q \mid q \text{ can be reached from } R \text{ with 0 or more } \varepsilon \text{ transitions}\} \text{ for } R \subseteq Q.$$

- Then $q_0' = E(\{q_0\})$.

NFA=DFA

- Since an NFA accepts an input if *any* of the possible computations ends in an accept state, $F' = \{R \subseteq Q \mid R \text{ contains a state } r \in F\}$.

- **Almost done!** But we still need to take ε transitions into account. To that end, let

$$E(R) = \{q \mid q \text{ can be reached from } R \text{ with 0 or more } \varepsilon \text{ transitions}\} \text{ for } R \subseteq Q.$$

- Then $q_0' = E(\{q_0\})$.
- Transition function δ':

$$\begin{aligned}\delta'(R, w_i) &= \bigcup_{r \in R} E(\delta(r, w_i)) \\ &= \{q \in Q \mid q \in E(\delta(r, w_i)) \text{ for some } r \in R\}\end{aligned}$$

NFA=DFA

In other words, we have just proven:

Theorem

A language is regular iff (if and only if) there exists an NFA that accepts it.

Concatenation of Regular Languages

- Remember from the previous lecture we proved that RL is closed under union and intersection?
- Proof method from the previous lecture:

  we could construct DFA for $L_1 \cup L_2$ and for $L_1 \cap L_2$ from DFAs for L_1 and L_2

- What about concatenation of languages?

Concatenation of Regular Languages

- Remember from the previous lecture we proved that RL is closed under union and intersection?
- Proof method from the previous lecture:

  we could construct DFA for $L_1 \cup L_2$ and for $L_1 \cap L_2$ from DFAs for L_1 and L_2
- What about concatenation of languages?
- Given two regular languages L_1, L_2, is

$$L_1 L_2 = \{w_1 w_2 \mid w_1 \in L_1, w_2 \in L_2\}$$

  a regular language?
- Can we use a similar proof method for concatenation (and Kleene star)?

Concatenation of Regular Languages

- Remember from the previous lecture we proved that RL is closed under union and intersection?
- Proof method from the previous lecture:

  we could construct DFA for $L_1 \cup L_2$ and for $L_1 \cap L_2$ from DFAs for L_1 and L_2
- What about concatenation of languages?
- Given two regular languages L_1, L_2, is

$$L_1 L_2 = \{w_1 w_2 \mid w_1 \in L_1, w_2 \in L_2\}$$

  a regular language?
- Can we use a similar proof method for concatenation (and Kleene star)?
- We can do this by constructing NFAs!

Concatenation of Regular Languages

- **Idea:** An automaton gets an input w and must "guess" how to split it $w = w_1 w_2$ in order to check L_1 and L_2 membership.

Concatenation of Regular Languages

- **Idea:** An automaton gets an input w and must "guess" how to split it $w = w_1 w_2$ in order to check L_1 and L_2 membership.

- **Example:**
  $L_1 = \{w \mid w \text{ begins and ends with } 1\}$ and $L_2 = \{w \mid w \text{ begins and ends with } 0\}$

  How to split the word $w = 10101010$? Multiple options

Concatenation of Regular Languages

- **Idea:** An automaton gets an input w and must "guess" how to split it $w = w_1 w_2$ in order to check L_1 and L_2 membership.

- **Example:**
  $L_1 = \{w \mid w \text{ begins and ends with } 1\}$ and $L_2 = \{w \mid w \text{ begins and ends with } 0\}$

  How to split the word $w = 10101010$? Multiple options

- This "guessing" is done by using nondeterminism:
  NFAs try all computational branches and accept if one leads to an accepting state.

Concatenation of Regular Languages

- **Idea:** An automaton gets an input w and must "guess" how to split it $w = w_1 w_2$ in order to check L_1 and L_2 membership.

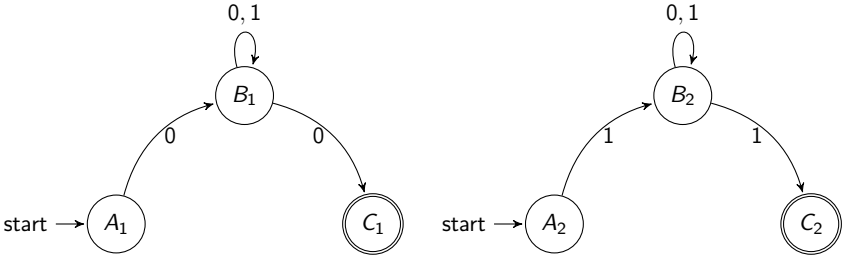
- **Example:**
  $L_1 = \{w \mid w \text{ begins and ends with } 1\}$ and $L_2 = \{w \mid w \text{ begins and ends with } 0\}$

  How to split the word $w = 10101010$? Multiple options

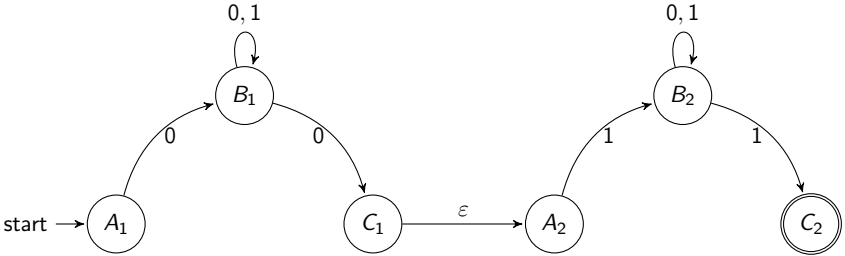
- This "guessing" is done by using nondeterminism:
  NFAs try all computational branches and accept if one leads to an accepting state.

- For concatenation: after M_1 has parsed a part of the input that led to an accept state, nondeterministically "jump" to the second automaton M_2.

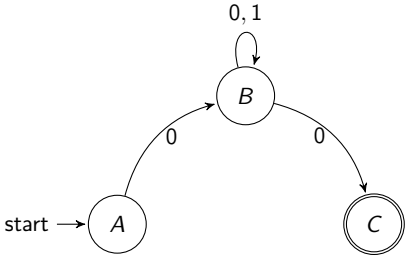
Concatenation of Regular Languages



Concatenation of Regular Languages

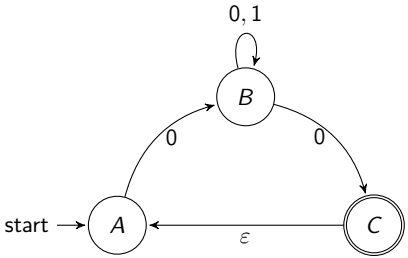


Kleene Star



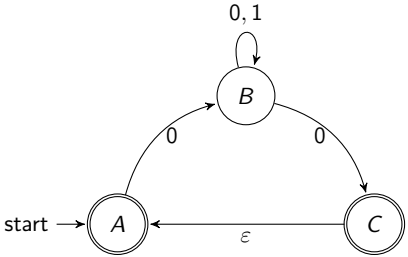
- Let L be the language recognized by this automaton
- How can we construct an automaton that recognizes L^*?

Kleene Star



- Let L be the language recognized by this automaton
- How can we construct an automaton that recognizes L^*?

Kleene Star



- Let L be the language recognized by this automaton
- How can we construct an automaton that recognizes L^*?

Summary

- **Basics**: sets, de Morgan's laws, tuples, products, functions, strings
- **Deterministic finite automata (DFA)**: states, transitions, start state, accept states
- Formal definitions of DFA, acceptance, regular language (RL)
- Operations on languages: union, intersection, concatenation, Kleene star
- Theorem: RL closed under union and intersection
- **Nondeterministic finite automata (NFA)**
- NFAs recognize regular languages (just like DFAs)
- Theorem: Every NFA has an equivalent DFA
- NFAs are convenient for composing languages (concatenation, repetition)
- Theorem: RL closed under concatenation and Kleene star