IN2080

3. Regular Expressions and Nonregular languages

Einar Broch Johnsen

University of Oslo

7th February 2023



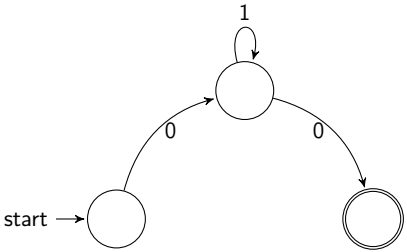
Department of Informatics



University of Oslo

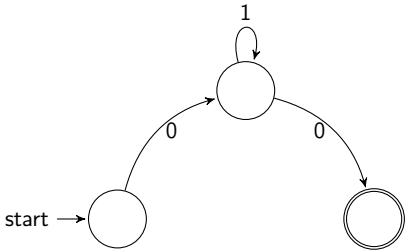
Last week

- Deterministic finite automata (DFA)



Last week

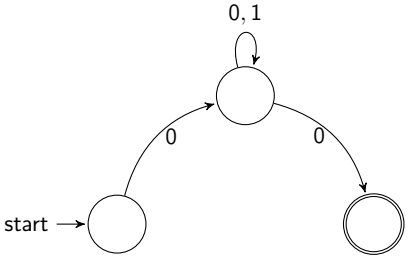
- Deterministic finite automata (DFA)

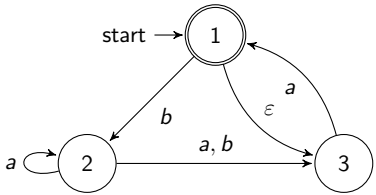


- Regular languages are those languages accepted by DFAs

Last week

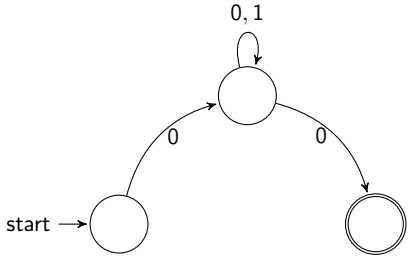
- Nondeterministic automata (NFA)

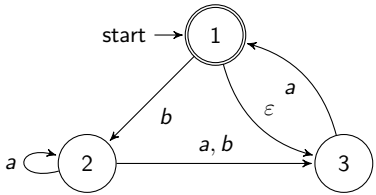




Last week

- Nondeterministic automata (NFA)





- DFA $\leftrightarrow$ NFA

Last week

- To show that a language is regular, we can construct a DFA *or* NFA that accepts the language

Last week

- To show that a language is regular, we can construct a DFA *or* NFA that accepts the language
- RL closed under union, intersection, concatenation, Kleene star operations

Last week

- To show that a language is regular, we can construct a DFA *or* NFA that accepts the language
- RL closed under union, intersection, concatenation, Kleene star operations
- Union, intersection: showed that we could construct DFA for $L_1 \cup L_2$ and $L_1 \cap L_2$ from DFAs for L_1 and L_2

Last week

- To show that a language is regular, we can construct a DFA *or* NFA that accepts the language
- RL closed under union, intersection, concatenation, Kleene star operations
- Union, intersection: showed that we could construct DFA for $L_1 \cup L_2$ and $L_1 \cap L_2$ from DFAs for L_1 and L_2
- Concatenation, Kleene star: showed that we could construct NFA for $L_1 L_2$ and L_1^* from NFAs for L_1 and L_2

Main topic today

- Regular expressions

Regular Expressions

We give an inductive definition of REs

Definition (Regular Expression)

Given an alphabet Σ, a *regular expression* is

- a for some $a \in \Sigma$,

Regular Expressions

We give an inductive definition of REs

Definition (Regular Expression)

Given an alphabet Σ, a *regular expression* is

- a for some $a \in \Sigma$,
- ε,

Regular Expressions

We give an inductive definition of REs

Definition (Regular Expression)

Given an alphabet Σ, a *regular expression* is

- a for some $a \in \Sigma$,
- ε,
- $\emptyset$,

Regular Expressions

We give an inductive definition of REs

Definition (Regular Expression)

Given an alphabet Σ, a *regular expression* is

- a for some $a \in \Sigma$,
- ε,
- $\emptyset$,
- $(R_1 \cup R_2)$ for regular expressions R_1, R_2,

Regular Expressions

We give an inductive definition of REs

Definition (Regular Expression)

Given an alphabet Σ, a *regular expression* is

- a for some $a \in \Sigma$,
- ε,
- $\emptyset$,
- $(R_1 \cup R_2)$ for regular expressions R_1, R_2,
- $(R_1 R_2)$ for regular expressions R_1, R_2,

Regular Expressions

We give an inductive definition of REs

Definition (Regular Expression)

Given an alphabet Σ, a *regular expression* is

- a for some $a \in \Sigma$,
- ε,
- $\emptyset$,
- $(R_1 \cup R_2)$ for regular expressions R_1, R_2,
- $(R_1 R_2)$ for regular expressions R_1, R_2,
- R_1^* for a regular expression R_1.

Regular Expressions

We give an inductive definition of REs

Definition (Regular Expression)

Given an alphabet Σ, a *regular expression* is

- a for some $a \in \Sigma$,
- ε,
- $\emptyset$,
- $(R_1 \cup R_2)$ for regular expressions R_1, R_2,
- $(R_1 R_2)$ for regular expressions R_1, R_2,
- R_1^* for a regular expression R_1.

$\rightarrow$ Regular expressions represent languages!

Regular Expressions - Examples

What languages do the following regular expressions (RE) represent?

- 0^*

Regular Expressions - Examples

What languages do the following regular expressions (RE) represent?

- 0^*
- 10^*1

Regular Expressions - Examples

What languages do the following regular expressions (RE) represent?

- 0^*
- 10^*1
- $(1(0 \cup 1)^*1) \cup (0(0 \cup 1)^*0) \cup 0 \cup 1$

Regular Expressions - Automata

What is the connection between RE and DFA/NFA?

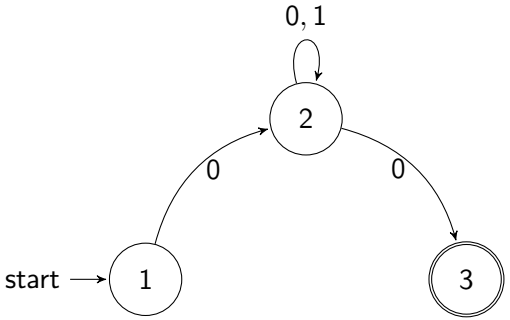
Regular Expressions - Automata

What is the connection between RE and DFA/NFA?

Language $0(0 \cup 1)^*0$:

What is the connection between RE and DFA/NFA?

Language $0(0 \cup 1)^*0$:



What is the connection between RE and DFA/NFA?

- Can all RE be represented using DFA/NFA?
- Can all DFA/NFA be described by RE?

What is the connection between RE and DFA/NFA?

- Can all RE be represented using DFA/NFA?
- Can all DFA/NFA be described by RE?

Yes!

Proposition

Every language described by an RE is regular.

Proof based on inductive definition of RE!

RE Describes Regular Languages

Definition (Regular Expression)

Given an alphabet Σ, a *regular expression* is

- a for some $a \in \Sigma$,
- ε,
- $\emptyset$,
- $(R_1 \cup R_2)$ for regular expressions R_1, R_2,
- $(R_1 R_2)$ for regular expressions R_1, R_2,
- R_1^* for a regular expression R_1.

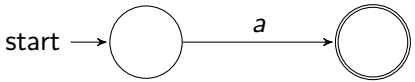
RE Describes Regular Languages

Definition (Regular Expression)

Given an alphabet Σ, a *regular expression* is

- a for some $a \in \Sigma$,
- ε,
- $\emptyset$,
- $(R_1 \cup R_2)$ for regular expressions R_1, R_2,
- $(R_1 R_2)$ for regular expressions R_1, R_2,
- R_1^* for a regular expression R_1.

If $R = a$ for $a \in \Sigma$, then $L(R) = \{a\}$ is accepted by



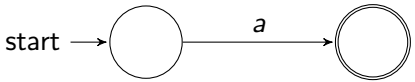
RE Describes Regular Languages

Definition (Regular Expression)

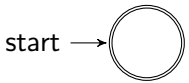
Given an alphabet Σ, a *regular expression* is

- a for some $a \in \Sigma$,
- ε,
- $\emptyset$,
- $(R_1 \cup R_2)$ for regular expressions R_1, R_2,
- $(R_1 R_2)$ for regular expressions R_1, R_2,
- R_1^* for a regular expression R_1.

If $R = a$ for $a \in \Sigma$, then $L(R) = \{a\}$ is accepted by



If $R = \varepsilon$, then $L(R) = \{\varepsilon\}$ is accepted by



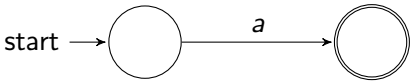
RE Describes Regular Languages

Definition (Regular Expression)

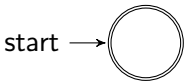
Given an alphabet Σ, a *regular expression* is

- a for some $a \in \Sigma$,
- ε,
- $\emptyset$,
- $(R_1 \cup R_2)$ for regular expressions R_1, R_2,
- $(R_1 R_2)$ for regular expressions R_1, R_2,
- R_1^* for a regular expression R_1.

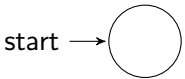
If $R = a$ for $a \in \Sigma$, then $L(R) = \{a\}$ is accepted by



If $R = \varepsilon$, then $L(R) = \{\varepsilon\}$ is accepted by



If $R = \emptyset$, then $L(R) = \emptyset$ is accepted by



RE Describes Regular Languages

Definition (Regular Expression)

Given an alphabet Σ, a *regular expression* is

- a for some $a \in \Sigma$,
- ε,
- $\emptyset$,
- $(R_1 \cup R_2)$ for regular expressions R_1, R_2,
- $(R_1 R_2)$ for regular expressions R_1, R_2,
- R_1^* for a regular expression R_1.

The rest is union, concatenation and Kleene star of regular languages, as discussed last week!

RE Describes Regular Languages

Definition (Regular Expression)

Given an alphabet Σ, a *regular expression* is

- a for some $a \in \Sigma$,
- ε,
- $\emptyset$,
- $(R_1 \cup R_2)$ for regular expressions R_1, R_2,
- $(R_1 R_2)$ for regular expressions R_1, R_2,
- R_1^* for a regular expression R_1.

The rest is union, concatenation and Kleene star of regular languages, as discussed last week!

(Recall: the union/concatenation/Kleene star of regular languages is itself regular)

We have just proven

Proposition

Every language described by a RE is regular.

We have just proven

Proposition

Every language described by a RE is regular.

Next:

Proposition

Every regular language can be described using a RE.

We have just proven

Proposition

Every language described by a RE is regular.

Next:

Proposition

Every regular language can be described using a RE.

How can we prove that?

We have just proven

Proposition

Every language described by a RE is regular.

Next:

Proposition

Every regular language can be described using a RE.

How can we prove that?

Proof idea:

- Define an algorithm that transforms a DFA into an RE
- For this purpose, we go via a new kind of automata: GNFA

GNFA

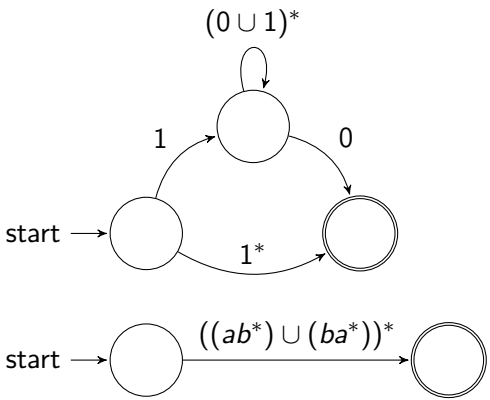
Generalized Nondeterministic Finite Automaton (GNFA):

- NFA where the transitions are RE, not only symbols from Σ.

GNFA

Generalized Nondeterministic Finite Automaton (GNFA):

- NFA where the transitions are RE, not only symbols from Σ.

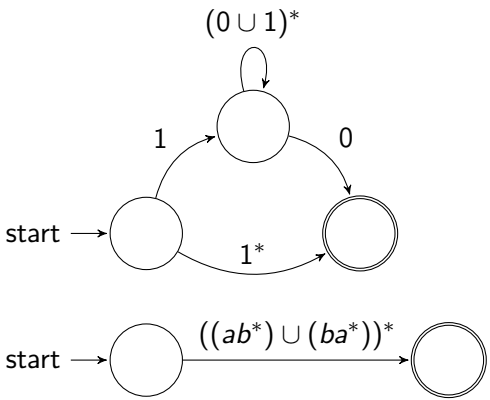


GNFA

Generalized Nondeterministic Finite Automaton (GNFA):

- NFA where the transitions are RE, not only symbols from Σ.

Some other assumptions for convenience:



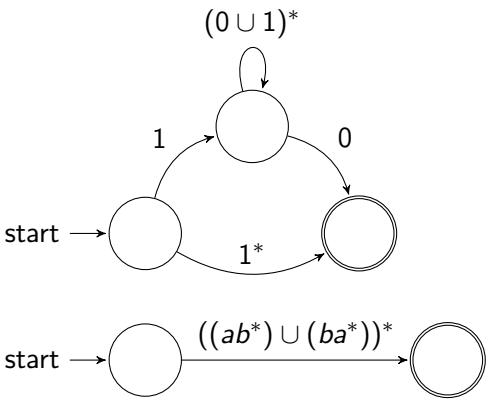
GNFA

Generalized Nondeterministic Finite Automaton (GNFA):

- NFA where the transitions are RE, not only symbols from Σ.

Some other assumptions for convenience:

- Start state goes to every other state, but has no incoming states



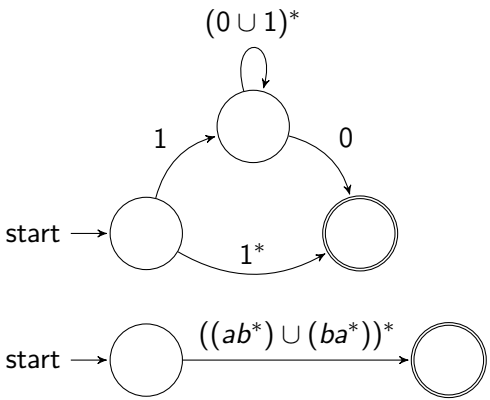
GNFA

Generalized Nondeterministic Finite Automaton (GNFA):

- NFA where the transitions are RE, not only symbols from Σ.

Some other assumptions for convenience:

- Start state goes to every other state, but has no incoming states
- Every state goes to the unique accepting state, which is different from the starting state. The accepting state does not have any outgoing arrows.



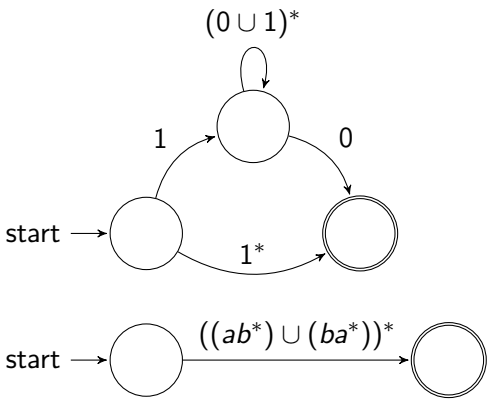
GNFA

Generalized Nondeterministic Finite Automaton (GNFA):

- NFA where the transitions are RE, not only symbols from Σ.

Some other assumptions for convenience:

- Start state goes to every other state, but has no incoming states
- Every state goes to the unique accepting state, which is different from the starting state. The accepting state does not have any outgoing arrows.
- All other states have one transition to all other states, including themselves.



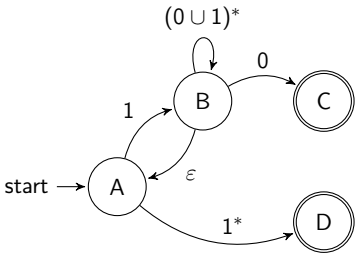
GNFA: Convenient assumptions

Generalized Nondeterministic Finite Automaton (GNFA):

- NFA where the transitions are RE, not only symbols from Σ.

Some other assumptions for convenience:

- Start state goes to every other state, but has no incoming states
- Every state goes to the unique accepting state, which is different from the starting state. The accepting state does not have any outgoing arrows.
- All other states have one transition to all other states, including themselves.



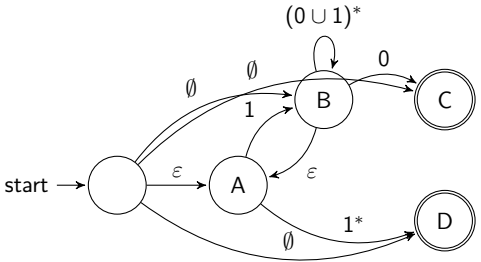
GNFA: Convenient assumptions

Generalized Nondeterministic Finite Automaton (GNFA):

- NFA where the transitions are RE, not only symbols from Σ.

Some other assumptions for convenience:

- Start state goes to every other state, but has no incoming states ✓
- Every state goes to the unique accepting state, which is different from the starting state. The accepting state does not have any outgoing arrows.
- All other states have one transition to all other states, including themselves.



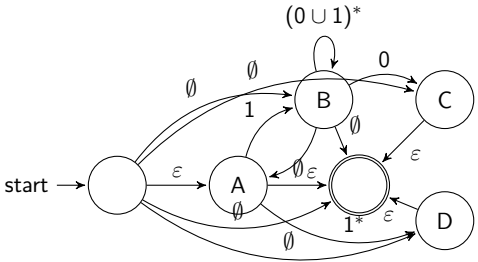
GNFA: Convenient assumptions

Generalized Nondeterministic Finite Automaton (GNFA):

- NFA where the transitions are RE, not only symbols from Σ.

Some other assumptions for convenience:

- Start state goes to every other state, but has no incoming states ✓
- Every state goes to the unique accepting state, which is different from the starting state. The accepting state does not have any outgoing arrows. ✓
- All other states have one transition to all other states, including themselves. (✓)



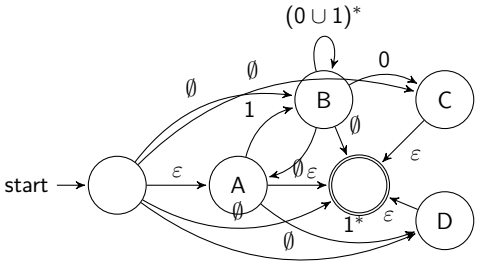
GNFA: Convenient assumptions

Generalized Nondeterministic Finite Automaton (GNFA):

- NFA where the transitions are RE, not only symbols from Σ.

Some other assumptions for convenience:

- Start state goes to every other state, but has no incoming states ✓
- Every state goes to the unique accepting state, which is different from the starting state. The accepting state does not have any outgoing arrows. ✓
- All other states have one transition to all other states, including themselves. (✓)



For the last point, add $\emptyset$ transitions between any two non-accepting/starting states that were not previously connected (e.g., (B, D))

Generalized Nondeterministic Finite Automata

Definition

A generalized nondeterministic finite automaton (GNFA) is a 5-tuple $(Q, \Sigma, \delta, q_{start}, q_{accept})$ where

1. Q is a finite set of states
2. Σ is the input alphabet
3. $\delta : (Q \setminus \{q_{accept}\}) \times (Q \setminus \{q_{start}\}) \to \mathcal{R}$ is the transition function, where $\mathcal{R}$ is the set of all RE's over Σ,
4. q_{start} is the start state, and
5. q_{accept} is the accept state.

Proposition

Every regular language can be described using a RE.

Proposition

Every regular language can be described using a RE.

Proof idea:

- Take DFA and transform into a GNFA that accepts the same language.
- Iteratively remove (non-starting and non-accepting) states so that the same language is accepted, until only the starting and accepting states remain.
- Then the RE along the transition between the two states describes the regular language.

Proposition

Every regular language can be described using a RE.

Proposition

Every regular language can be described using a RE.

Proof:

- Given a DFA M, we construct an equivalent GNFA G by adding a new start state q_{start} with an ε transition to the old start state q_0, as well as a new accepting state q_{accept}, with ε transitions from all old accept states.
- Add $\emptyset$ transitions for all state pairs that do not have a transition in M.

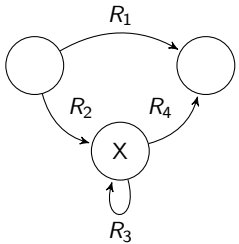
Recall the "convenient" properties of GNFA:

- Start state goes to every other state, but has no incoming states
- Every state goes to the unique accepting state, which is different from the starting state. The accepting state does not have any outgoing arrows.
- All other states have one transition to all other states, including themselves.

Recall the "convenient" properties of GNFA:

- Start state goes to every other state, but has no incoming states
- Every state goes to the unique accepting state, which is different from the starting state. The accepting state does not have any outgoing arrows.
- All other states have one transition to all other states, including themselves.

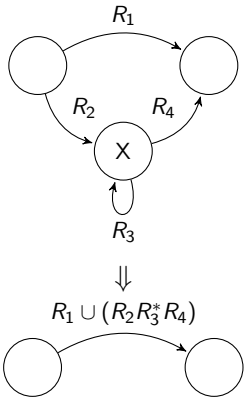
$\Rightarrow$ When removing X, we only need to consider situations like this:



Recall the "convenient" properties of GNFA:

- Start state goes to every other state, but has no incoming states
- Every state goes to the unique accepting state, which is different from the starting state. The accepting state does not have any outgoing arrows.
- All other states have one transition to all other states, including themselves.

$\Rightarrow$ When removing X, we only need to consider situations like this:



Let's formalize this!

Let $G = (Q, \Sigma, \delta, q_{start}, q_{accept})$ with $k = |Q|$.

Let's formalize this!

Let $G = (Q, \Sigma, \delta, q_{start}, q_{accept})$ with $k = |Q|$.
Define a procedure CONVERT(G):

1. If $k = 2$ then G only has one start and one accept state.
   Return the regular expression R of the transition connecting them.

Let's formalize this!

Let $G = (Q, \Sigma, \delta, q_{start}, q_{accept})$ with $k = |Q|$.
Define a procedure CONVERT(G):

1. If $k = 2$ then G only has one start and one accept state.
   Return the regular expression R of the transition connecting them.
2. If $k > 2$ select a state $q' \notin \{q_{accept}, q_{start}\}$.
   Define $G' = \{Q', \Sigma, \delta', q_{start}, q_{accept}\}$ with $Q' = Q \setminus \{q'\}$ and

$$\delta'(q_i, q_j) = R_1 \cup R_2 R_3^* R_4$$

   where $R_1 = \delta(q_i, q_j)$, $R_2 = \delta(q_i, q')$, $R_3 = \delta(q', q')$, $R_4 = \delta(q', q_j)$.

Let's formalize this!

Let $G = (Q, \Sigma, \delta, q_{start}, q_{accept})$ with $k = |Q|$.
Define a procedure CONVERT(G):

1. If $k = 2$ then G only has one start and one accept state.
   Return the regular expression R of the transition connecting them.
2. If $k > 2$ select a state $q' \notin \{q_{accept}, q_{start}\}$.
   Define $G' = \{Q', \Sigma, \delta', q_{start}, q_{accept}\}$ with $Q' = Q \setminus \{q'\}$ and

$$\delta'(q_i, q_j) = R_1 \cup R_2 R_3^* R_4$$

   where $R_1 = \delta(q_i, q_j)$, $R_2 = \delta(q_i, q')$, $R_3 = \delta(q', q')$, $R_4 = \delta(q', q_j)$.
3. Return the result of CONVERT(G').

Let's formalize this!

Let $G = (Q, \Sigma, \delta, q_{start}, q_{accept})$ with $k = |Q|$.
Define a procedure CONVERT(G):

1. If $k = 2$ then G only has one start and one accept state.
   Return the regular expression R of the transition connecting them.
2. If $k > 2$ select a state $q' \notin \{q_{accept}, q_{start}\}$.
   Define $G' = \{Q', \Sigma, \delta', q_{start}, q_{accept}\}$ with $Q' = Q \setminus \{q'\}$ and

$$\delta'(q_i, q_j) = R_1 \cup R_2 R_3^* R_4$$

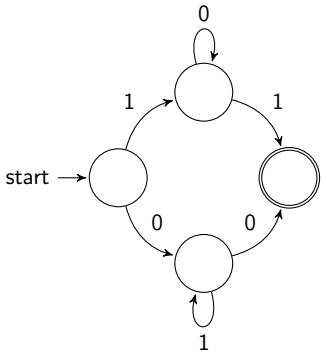
   where $R_1 = \delta(q_i, q_j)$, $R_2 = \delta(q_i, q')$, $R_3 = \delta(q', q')$, $R_4 = \delta(q', q_j)$.
3. Return the result of CONVERT(G').

Correctness still remains to be shown! See book for details! (Claim 1.65)

Proposition

Every regular language can be described using a RE.

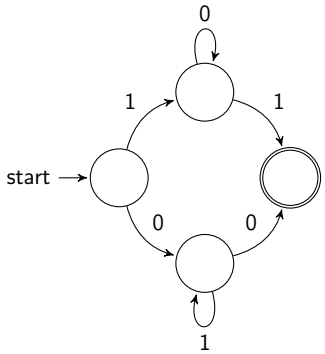
Example:
DFA:



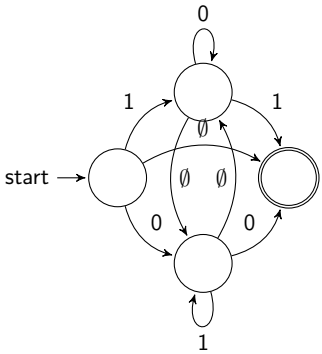
Proposition

Every regular language can be described using a RE.

Example:
DFA:



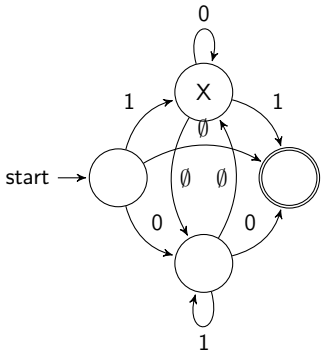
GNFA:



Proposition

Every regular language can be described using a RE.

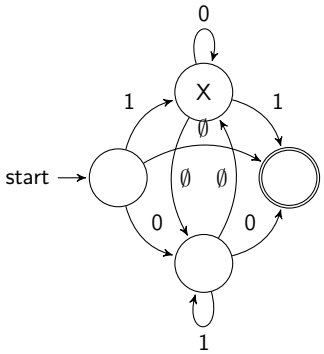
Example:
Remove state X:

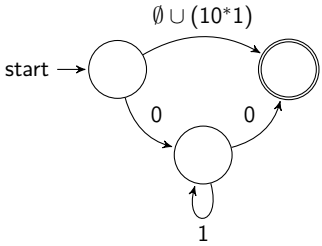


Proposition

Every regular language can be described using a RE.

Example:
Remove state X:

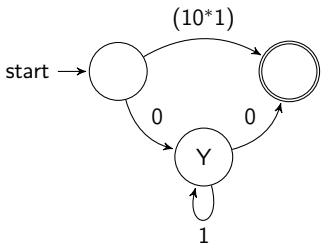




Proposition

Every regular language can be described using a RE.

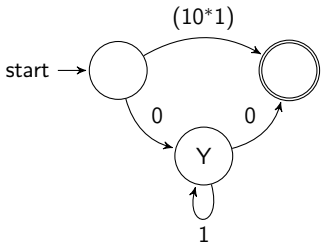
Example:
Remove state Y:

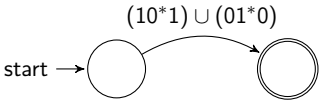


Proposition

Every regular language can be described using a RE.

Example:
Remove state Y:





Summary

So RE = GNFA = DFA = NFA = Regular languages...

Summary

So RE = GNFA = DFA = NFA = Regular languages...

But when is a language *nonregular*? How can we check?

Summary

So RE = GNFA = DFA = NFA = Regular languages...

But when is a language *nonregular*? How can we check?

$\Rightarrow$ Pumping Lemma!