Finite-State Automata

Yo-Sub Han
CS, Yonsei University

Overview of Unit

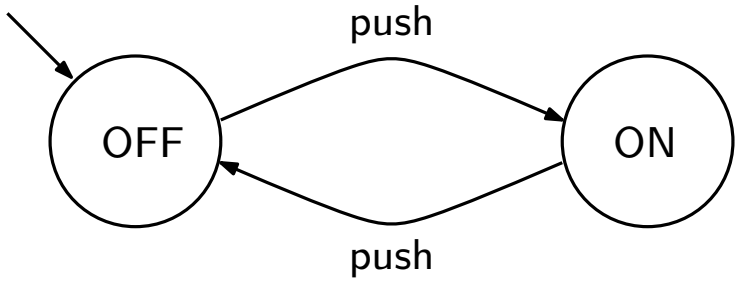
- Finite-state automata (FAs)
- Deterministic finite-state automata (DFAs) and configurations
- Nondeterministic finite-state automata (NFAs) and configurations
- λ-NFAs

Applications of FAs

- designing and checking the behavior of digital circuits
- lexical analyzer of a compiler
- scanning large boides of text such as web pages, source codes, bio data, BIG DATA to find occurrences of words, phrases or other pattens
- verifying systems of all types that have a finite number of distinct states

Switch

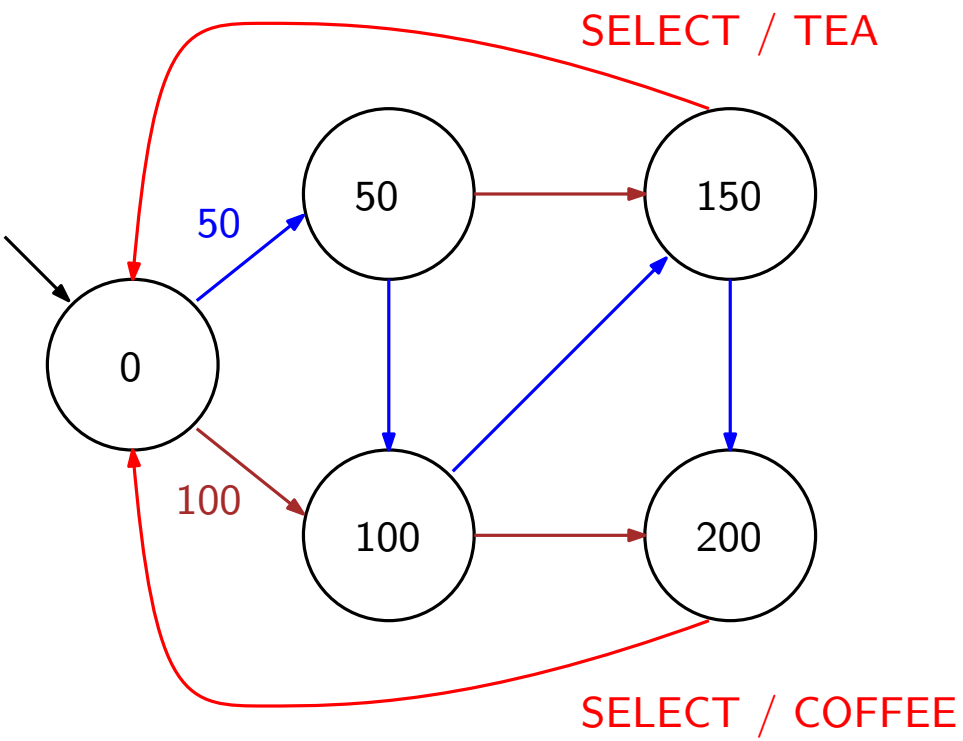
ON / OFF switch



ACTION:

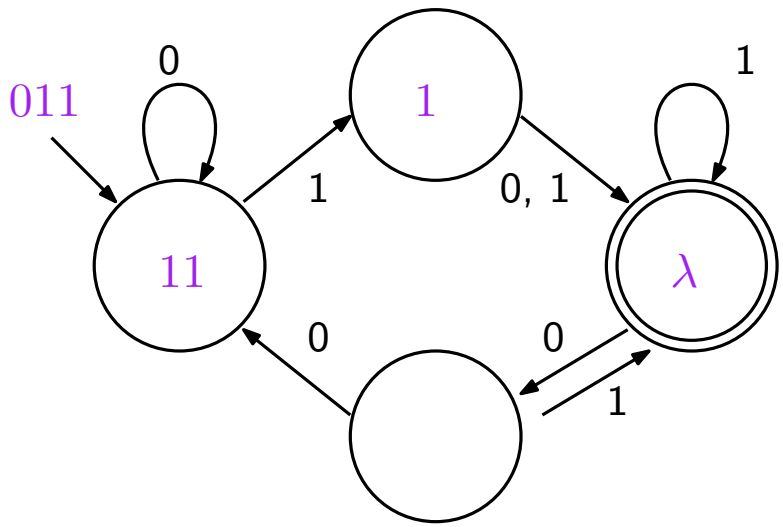
push	push	push	push	push

Vending Machine



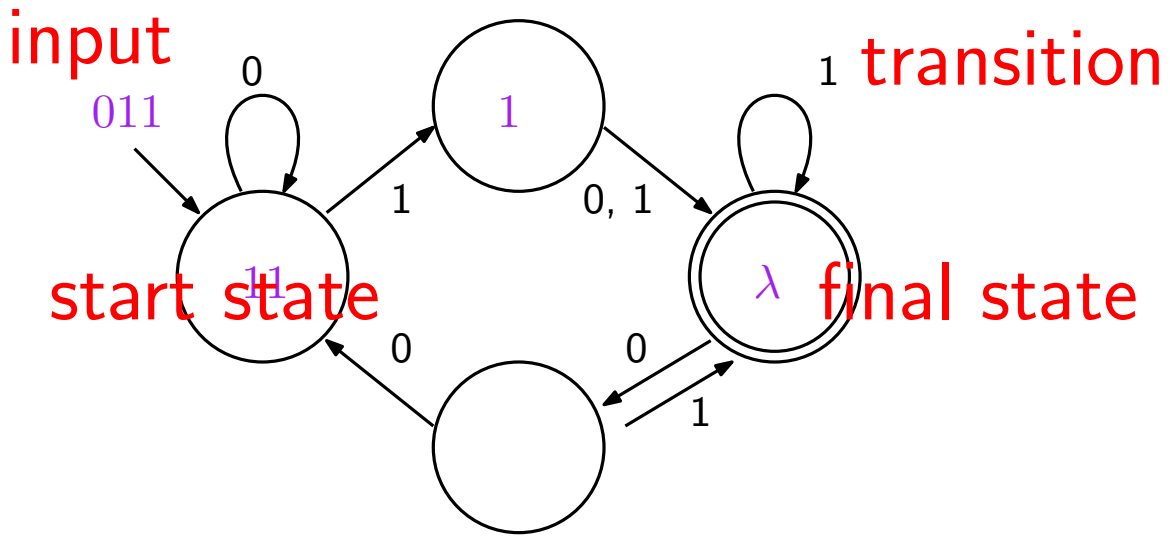
A vending machine for tea (150) and coffee (200)

A bit more serious example



The automaton accepts a string if it reaches a final state.

A bit more serious example



The automaton accepts a string if it reaches a final state.

Finite-State Automata (FAs)

- regular languages
- a set of finite states
- control moves from state to state in response to external "inputs"

Deterministic Finite-State Automata (DFAs)

- Deterministic: One and only one possible move
- Finite-State: A finite number of states
  - State: *condition* with respect to structure, from, constitutioin, phase and so on
- Automata: Machines

DFA

A DFA A is specified by a tuple $(Q, \Sigma, \delta, s, F)$, where

- Q is a finite, nonempty set of states
- Σ is an input alphabet
- δ is a transition function $Q \times \Sigma \to Q$. We capture the meaning of the function schematically as follows:

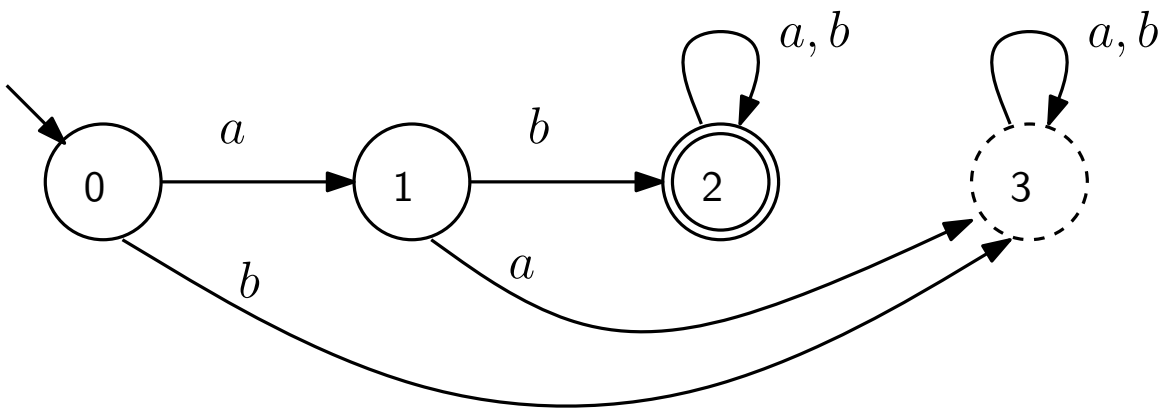
Current State	Current Character	Next State
p	a	$\delta(p, a) = q$

- s is the start state
- F is a set of final states

DFA Example

A DFA A is specified by $Q = \{0, 1, 2, 3\}$, $\Sigma = \{a, b\}$, $s = 0$, $F = \{2\}$ and δ is given as follows:

	a	b
0	1	3
1	3	2
2	2	2
3	3	3



Transition graph for A

How a DFA Works

Given a sequence w of input symobls:

$$w = a_1 a_2 a_3 \cdots a_n \in \Sigma^*.$$

1. Start from s in the DFA
2. Find the next state q_1 from $\delta(s, a_1)$
3. Process the next symbol a_2
4. Repeat the previous step and find $q_2, q_3, \ldots, q_n$
   - Accept, if $q_n \in F$
   - Reject, otherwise

A DFA A is specified by a tuple $(Q, \Sigma, \delta, s, F)$, where

- Q is a finite, nonempty set of states
- Σ is an input alphabet
- δ is a transition function $Q \times \Sigma \to Q$.

Current State	Current Character	Next State
p	a	$\delta(p, a) = q$

- s is the start state
- F is a set of final states

DFA Specification: Example

A DFA A is specified by $Q = \{0, 1, 2, 3\}$, $\Sigma = \{a, b\}$, $s = 0$, $F = \{2\}$ and δ is given as follows:

	a	b
0	1	3
1	2	3
2	2	2
3	3	3

Example:

- $w_1 = aaa$
- $w_2 = abb$

DFA Configuration

- A configuration captures the current automaton status. If an automaton crashes and we know its current status, we can restart the automaton where it left off by using the latest configuration. For DFAs, we need only the current state and the current position of the reader in the input string.
- We use (current state, remainder of input strings), which is an element of $Q \times \Sigma^*$
- Example of DFA configuration:
  - $(q_3, abbcca)$
  - (q_7, cca)

Single-Step Computations

We now formalize the notion of a single-step computation.

1. One possible start configuration for A is $(0, aabba)$
2. Then, the configuration after one computational step is $(0, abba)$
3. What is the configuration after two computational steps?

	a	b
0	0	1
1	1	0

$s = 0, F = \{0\}$

We write these steps more formally as follows:

$$(0, aabba) \vdash_A (0, abba) \vdash_A (0, bba).$$

Single-step computations in a DFA A:

1. Let (p, w) be a current configuration, where $w = \sigma x, \sigma \in \Sigma$
2. If $\delta(p, a) = q$, then the next configuration is (q, x)
3. We say that (p, w) yields (q, x) in one step. Namely, $(p, w) \vdash_A (q, x)$
4. $\vdash_A$: $Q \times \Sigma^* \to Q \times \Sigma^*$

Single-Step Computations

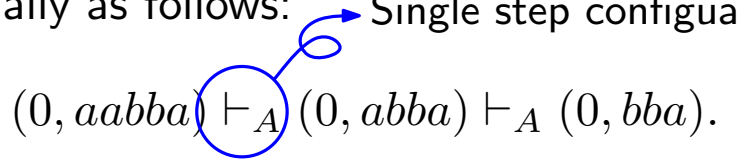
We now formalize the notion of a single-step computation.

1. One possible start configuration for A is $(0, aabba)$
2. Then, the configuration after one computational step is $(0, abba)$
3. What is the configuration after two computational steps?

	a	b
0	0	1
1	1	0

$s = 0, F = \{0\}$

We write these steps more formally as follows:



$$(0, aabba) \vdash_A (0, abba) \vdash_A (0, bba).$$

Single-step computations in a DFA A:

1. Let (p, w) be a current configuration, where $w = \sigma x, \sigma \in \Sigma$
2. If $\delta(p, a) = q$, then the next configuration is (q, x)
3. We say that (p, w) yields (q, x) in one step. Namely, $(p, w) \vdash_A (q, x)$
4. $\vdash_A$: $Q \times \Sigma^* \to Q \times \Sigma^*$

Multiple-Step Computations

We extend the single-step computation into multi(ple)-step computation as follows:

1. If $(p, w) \vdash_A (p_1, w_1) \vdash_A \cdots \vdash_A (p_{n-1}, w_{n-1}) \vdash_A (q, x)$, we say that (p, w) yields (q, x) in n steps

   $$(p, w) \vdash_A^n (q, x)$$

   or, more simply,

   $$(p, w) \vdash_A^* (q, x)$$

   in 0 or more steps

2. Every configuration yields itself in zero steps; namely $(p, w) \vdash_A^* (p, w)$

3. $\vdash_A^*$ is the reflexive and transitive closure of $\vdash_A$

4. We can also use $\vdash_A^+ (p, w)$

Acceptance and Language by a DFA

We define the acceptance of a string w with respect to a DFA A as follows:

1. We say w is accepted by A if and only if

$$(s, w) \vdash_A^* (f, \lambda) \text{ and } f \in F.$$

   In other words, there exists a sequence of configurations from the start state s to a final state f for the input string w with respect to A.

2. The language $L(A)$ of A is a set of all accepted strings;

$$L(A) = \{w \mid w \text{ is accepted by } A\}.$$

A DFA Example

Observe that the character "b" flips the current state in A whereas "a" does not.

$$\begin{aligned}(0, aabba) &\vdash_A (0, abba)\\ &\vdash_A (0, bba)\\ &\vdash_A (1, ba)\\ &\vdash_A (0, a)\\ &\vdash_A (0, \lambda)\end{aligned}$$

	a	b
0	0	1
1	1	0

$s = 0, F = \{0\}$

So, $(0, aabba) \vdash_A^* (0, \lambda)$ and, therefore, A accepts $aabba$.

$L(A) = \{w \mid w$ contains an even number of b's$\}$.

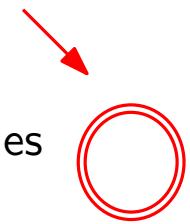
Transition Graphs

Transition graphs provide a schematic representation of a DFA.

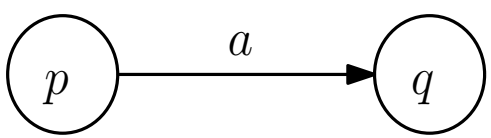
	a	b
0	0	1
1	1	0

$s = 0, F = \{0\}$

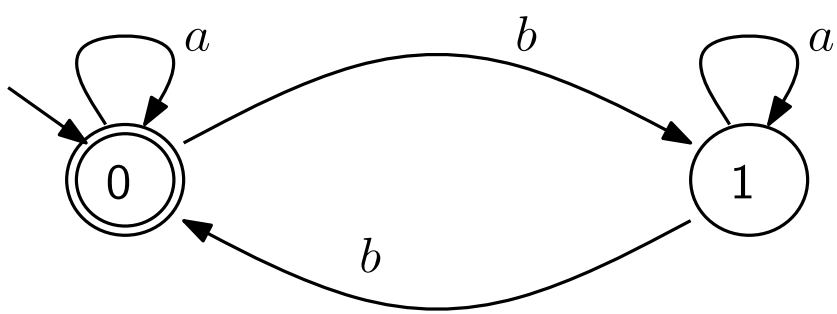
- Circles (Vertices) are states
  - The start state is indicated by a slant arrow
  - The final states are indicated by double circles



- Given a transition $\delta(p, a) = q$, there is an edge (transition) from p to q labeled with a:



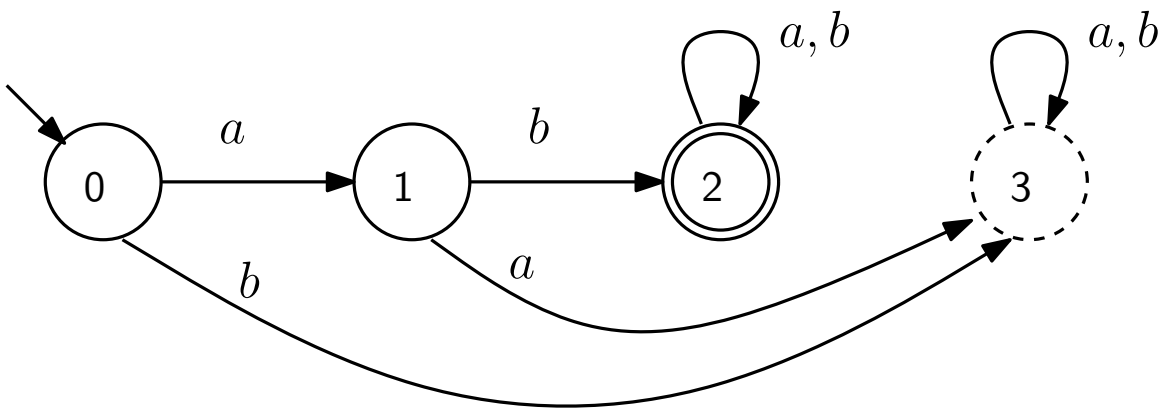
Transition graph for A



DFA Example

A DFA A is specified by $Q = \{0, 1, 2, 3\}$, $\Sigma = \{a, b\}$, $s = 0$, $F = \{2\}$ and δ is given as follows:

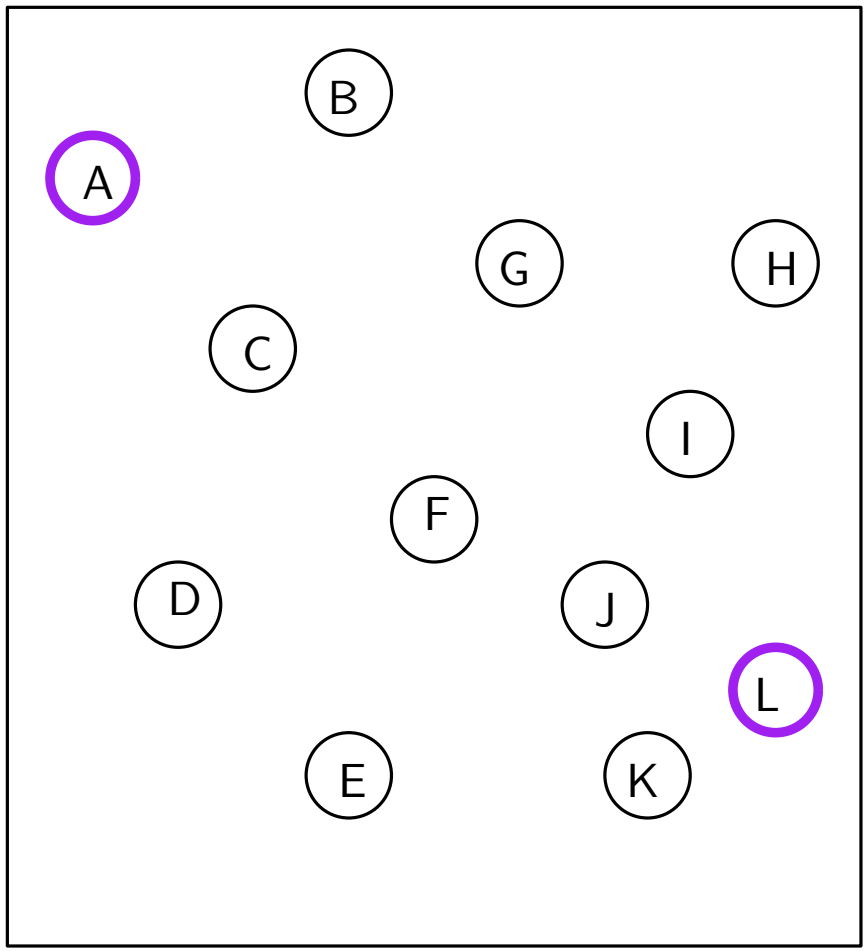
	a	b
0	1	3
1	3	2
2	2	2
3	3	3



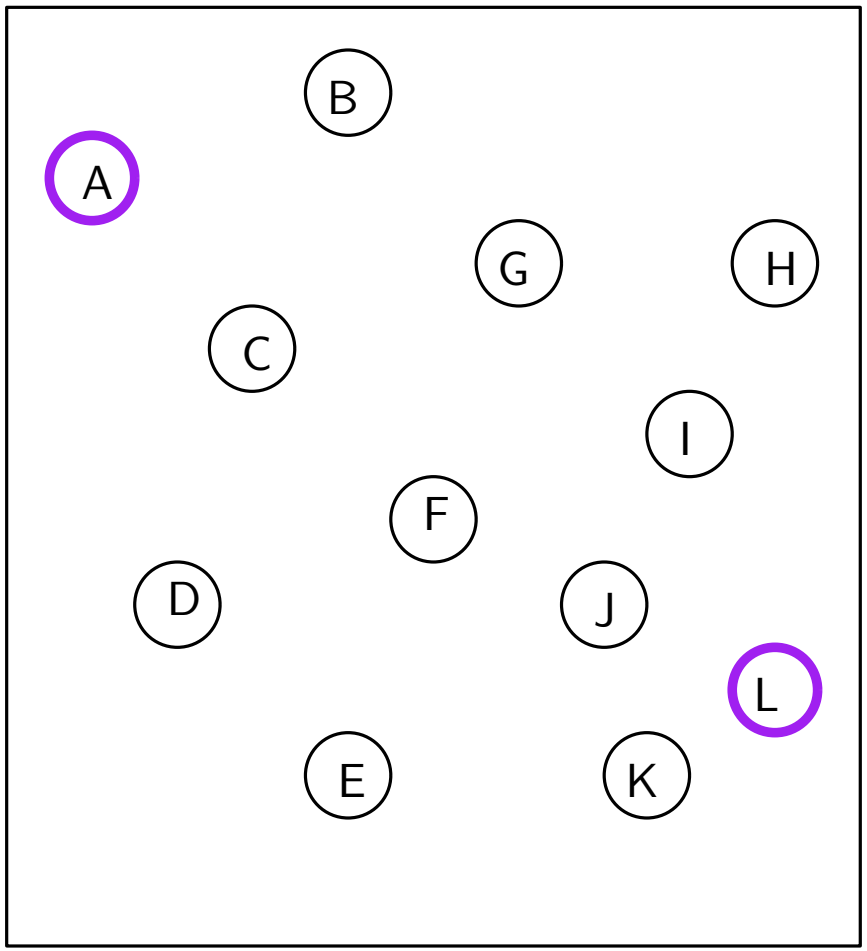
Transition graph for A

Determinism vs Nondeterminism

deterministic

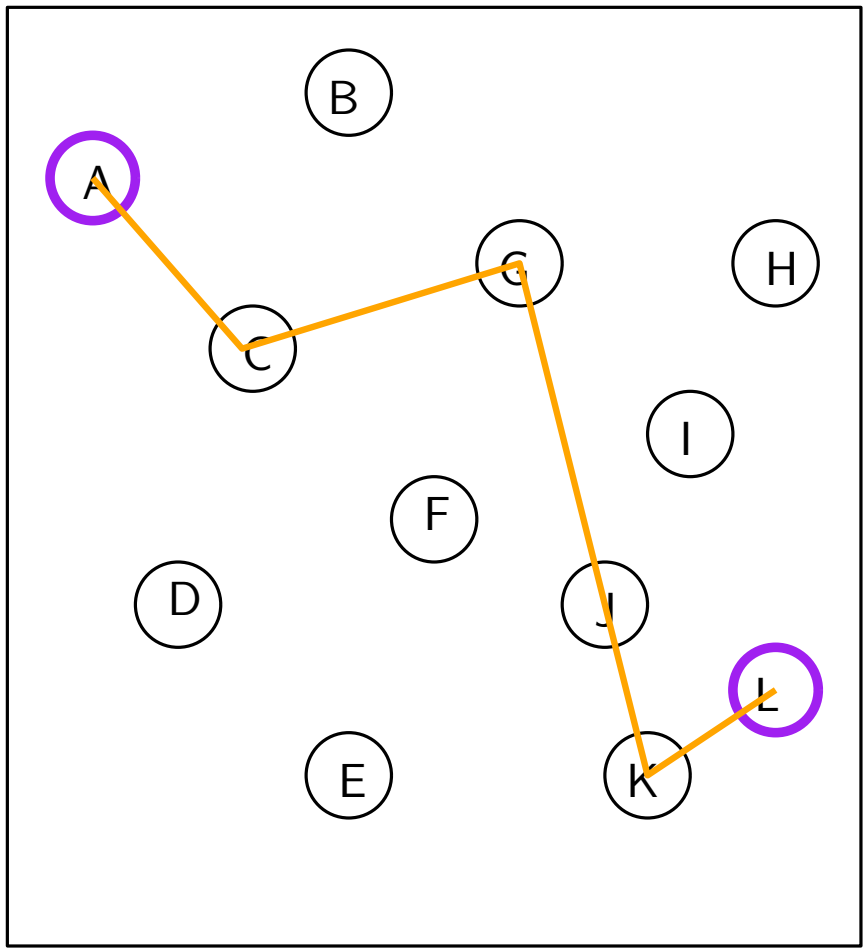


nondeterministic

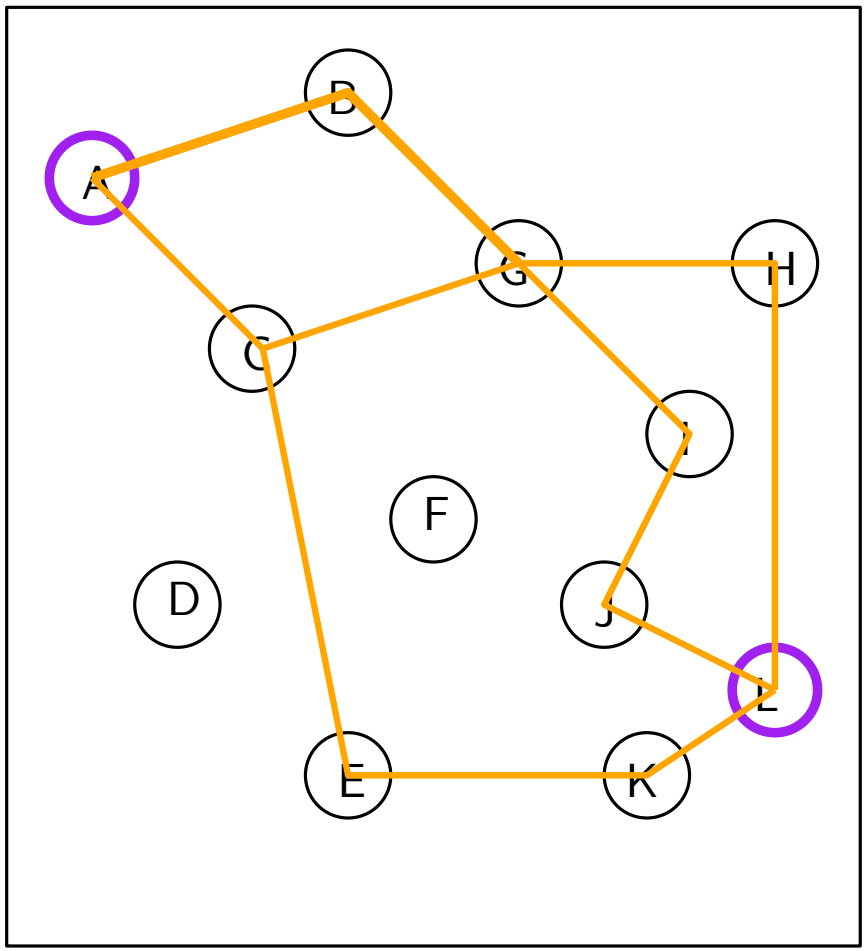


Determinism vs Nondeterminism

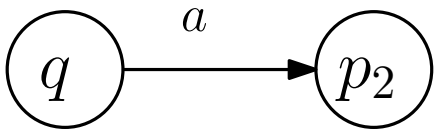
deterministic

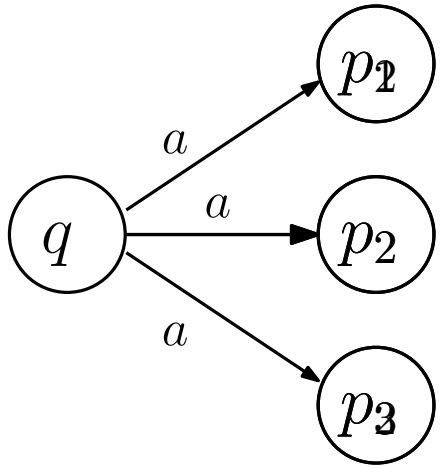


nondeterministic



DFAs and NFAs



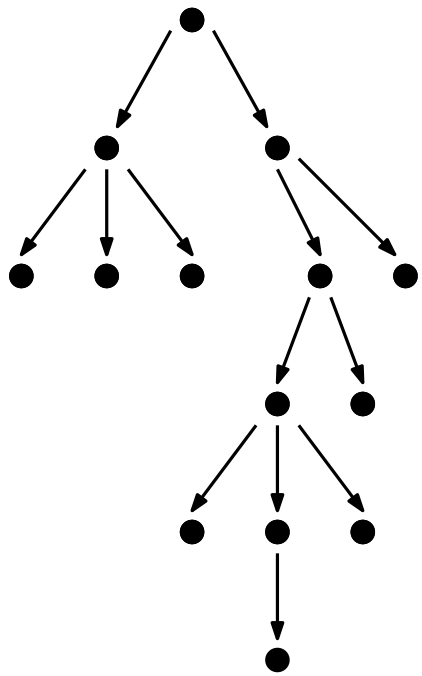


DFAs and NFAs

deterministic

accept or reject

nondeterministic



reject

accept

DFAs and NFAs

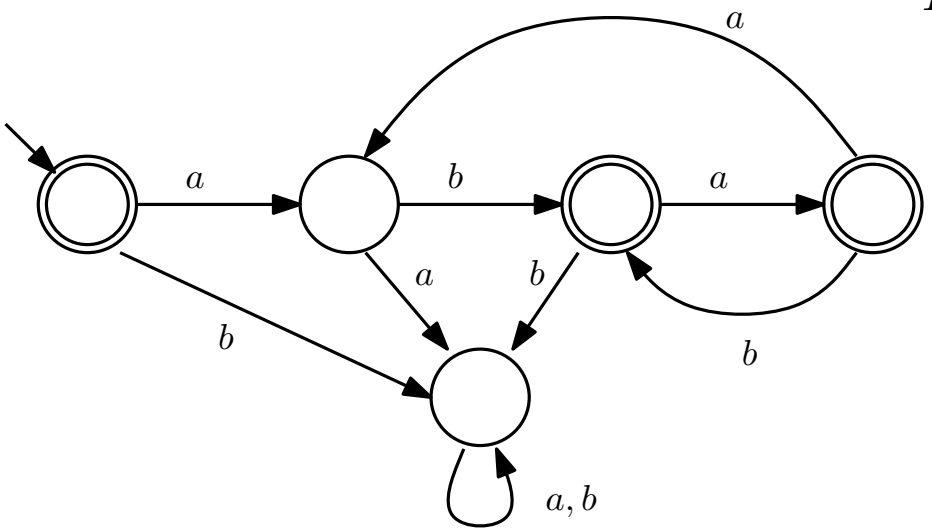
- In a DFA,
  1. The automaton has only one possible move for each character a state
  2. The next state is completely determined by the current state and current character (there is only one next state)
  3. Acceptance occurs if the computation reads all the input string and the computation ends at a final state

- In an NFA,
  1. The automaton may sometimes have more than one possible move
  2. The next state is only *"partially determined"* by the current state and input character (there may be two or more next states)
  3. Acceptance occurs if a computation reads all the input string and the computation ends at a final state

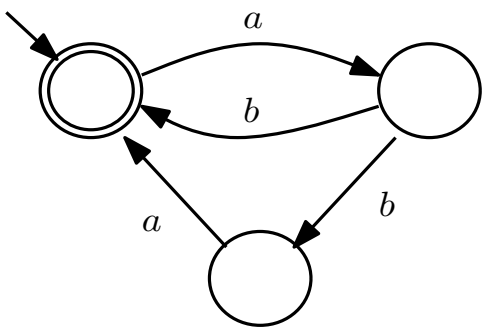
NFA Examples

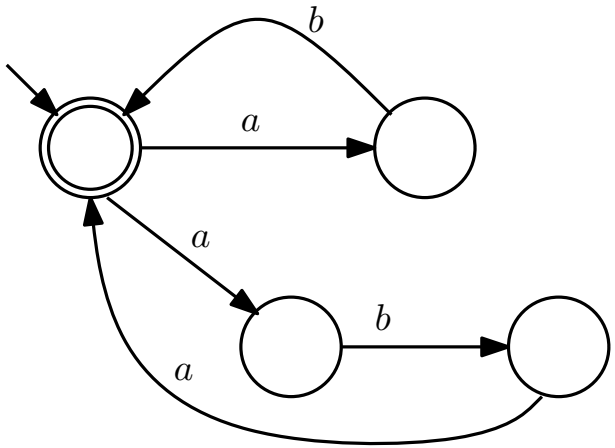
Consider the language L specified by the following DFA.

$$L = \{\lambda, (ab)^i, (aba)^i, ababa, abaab, \ldots\}$$



NFAs



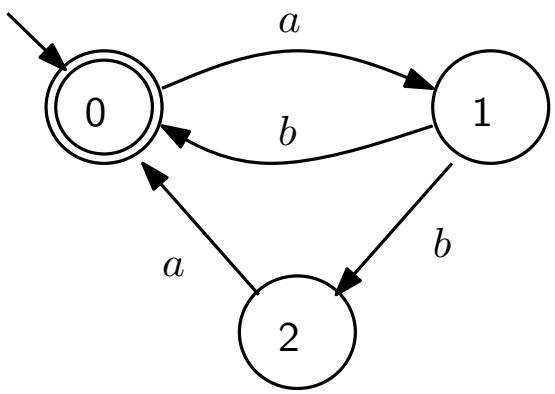


NFA Specification

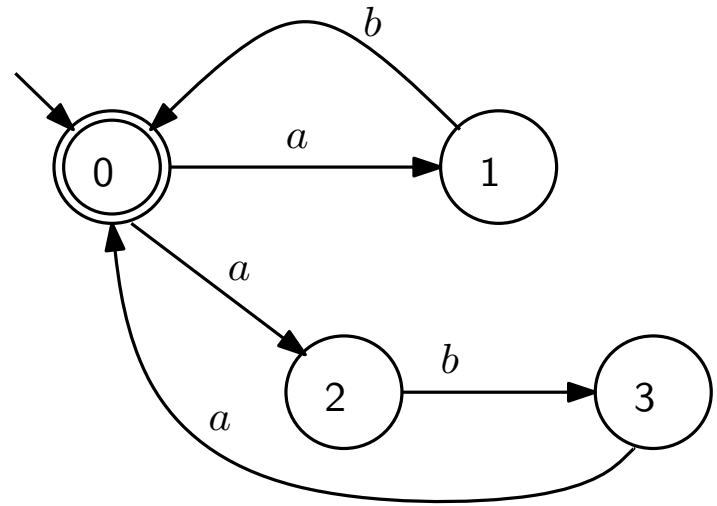
An NFA A is specified by a tuple $(Q, \Sigma, \delta, s, F)$, where

1. Q is a finite, nonempty set of states
2. Σ is an input alphabet
3. δ is a transition relation such that $\delta \subseteq Q \times \Sigma \times Q$.
   In other words, δ is a transition function $Q \times \Sigma \to Q^*$ (c.f., for DFA, $Q \times \Sigma \to Q$)
4. s is the start state
5. F is a set of final states

NFA Examples Revisited



(I)



(II)

- NFA (I): $Q = \{0, 1, 2\}, s = 0, F = \{0\}, \Sigma = \{a, b\}$ and $\delta = \{(0, a, 1), (1, b, 0), (1, b, 2), (2, a, 0)\}$.
  Note that $(1, b, 0)$ means that when the automaton is in state 1, it may consume b and enter state 0

- NFA (II): $Q = \{0, 1, 2, 3\}, s = 0, F = \{0\}, \Sigma = \{a, b\}$ and $\delta = \{(0, a, 1), (0, a, 2), (1, b, 0), (2, b, 3), (3, a, 0)\}$

NFA Computations

Given an NFA $A = (Q, \Sigma, \delta, s, F)$:

1. $(q, w) \vdash_A (q', w')$ if (and only if) there is a character $\sigma \in \Sigma$ such that $w = \sigma w'$ and (q, σ, q') is in δ
2. $\vdash_A^*$ is the reflexive and transitive closure of $\vdash_A$
3. A string w is accepted by A if and only if there is a computation

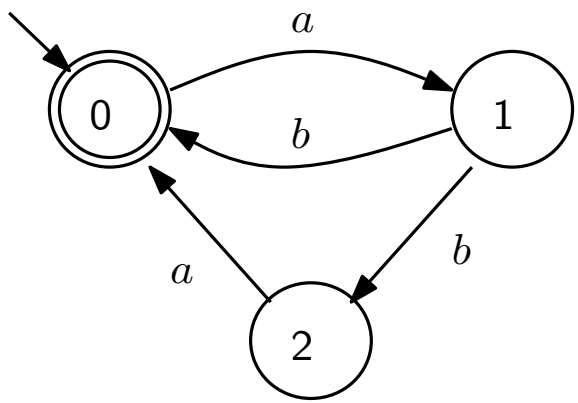
$$(s, w) \vdash_A^* (f, \lambda) \text{ and } f \in F.$$

4. The language $L(A)$ of A is defined as

$$L(A) = \{w \mid w \text{ is accepted by } A\}.$$

NFA Computations Example

Consider the NFA (I) specified as follows: $Q = \{0, 1, 2\}, s = 0, F = \{0\}, \Sigma = \{a, b\}$ and $\delta = \{(0, a, 1), (1, b, 0), (1, b, 2), (2, a, 0)\}$.



(I)

1. One computation of (I):
   $(0, aba) \vdash (1, ba) \vdash (0, a) \vdash (1, \lambda)$
   $(0, aba) \vdash^* (1, \lambda)$

2. Another computation of (I):
   $(0, aba) \vdash (1, ba) \vdash (2, a) \vdash (0, \lambda)$
   $(0, aba) \vdash^* (0, \lambda)$

NFA (I) accepts *aba* because of the second computation.
(Note that we have omitted the subscripts of $\vdash$ and $\vdash^*$ since we can assume that NFA (I) is nderstood.)

DFAs and NFAs

Between DFAs (deterministic) and NFAs (nondeterministic), which model have more expressive power? We know that, *by definition*, all DFAs are an NFA. Thus, NFAs are at least as powerful as DFAs.

We prove that every NFA $A = (Q, \Sigma, \delta, s, F)$ can be converted into an equivalent DFA $A' = (Q', \Sigma, \delta', s', F')$. (By equivalent, we mean that A and A' accept the same language.)

- Observation: Given A, for the same input string w, all computations for w read the same character at the same computational step (excep when some computations stop early)
- Idea: Combine all computations for w in A into one computation *that begins with the state set* $\{s\}$ and at each step computes the next state set, rather than the next state
- Construction: We compute all possible next state sets that are reachable from $\{s\}$, the SUBSET CONSTRUCTION.

Subset Construction

We compute all possible next state sets that are reachable from $\{s\}$ of an NFA A.

1. How many state sets are there, in the worst-case, for an NFA A with n states?
2. Which state sets are final state sets?
3. What is the new transition relation?

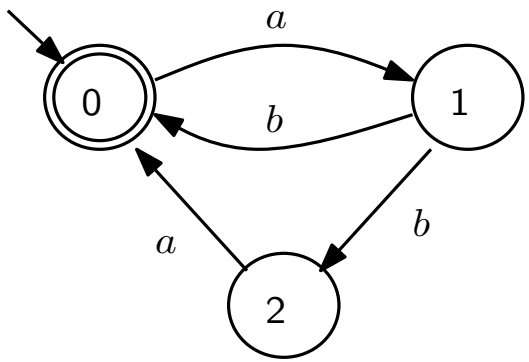
We construct δ' from δ as follows:

1. We construct transitions from $s' = \{s\}$ using the rule that $(s'\sigma, R)$ is in δ' of A' if and only if $R = \{r \mid (s, \sigma, r) \in \delta\}$
2. For each new state set P obtained in steps 1 and 2, we compute the transitions from P using the rule that (P, σ, R) is in δ' of A' if and only if $R = \{r \mid (p, \sigma, r) \in \delta \text{ and } p \in P\}$
3. When there are no new state sets generated in step 2, we have computed δ' and Q'

Now we define $F' = \{P \mid P \cap F \neq \emptyset\}$.

Subset Construction Example

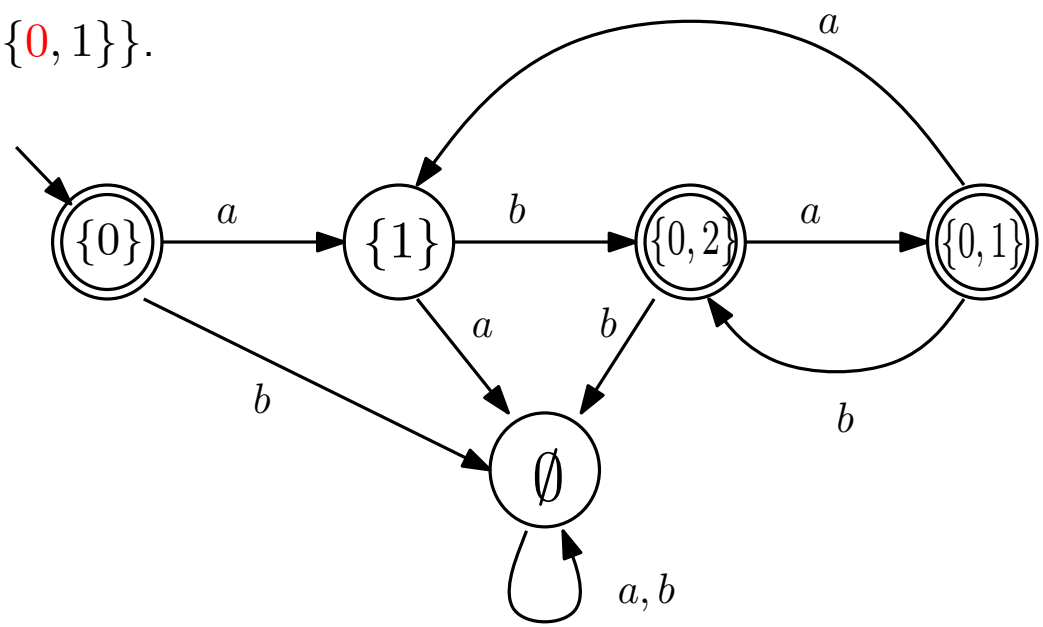
For the NFA (I), we have the following transition relation:



(I)

state set	a	b
$\{0\}$	$\{1\}$	$\emptyset$
$\{1\}$	$\emptyset$	$\{0,2\}$
$\{0,2\}$	$\{0,1\}$	$\emptyset$
$\{0,1\}$	$\{1\}$	$\{0,2\}$
$\emptyset$	$\emptyset$	$\emptyset$

Then, $F' = \{\{0\}, \{0,2\}, \{0,1\}\}$.



Subset Construction Correctness

Formally, we still have to prove the correctness of subset construction; however, we give only proof sketches. For the complete proof, see the textbook. Our proof of correctness has two parts:

1. Prove that A' is deterministic.
   This part is straightforward. Show that is (P, σ, R) and (P, σ, S) are both in δ', then $R = S$. Use the definition of the transitions for A'

2. Prove that $L(A') = L(A)$.
   This part is more complex. We need to prove, by induction on string length, that if $(s, w \vdash^* (f, \lambda)$ in A such that $f \in F$, then there is a $P \in F'$ such that $(\{s\}, w \vdash^* (P, \lambda)$ in A' and $f \in P$.

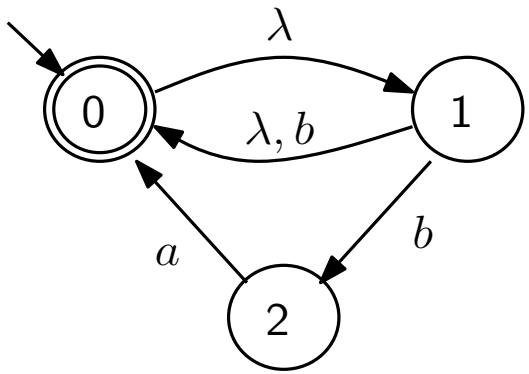
Therefore, NFAs have the same expressive power as DFAs.

λ-NFAs

A λ-NFA is an NFA that has a λ-transition.

A λ-transition

- allows to move between states without reading the input
- makes the FA highly nondeterministic

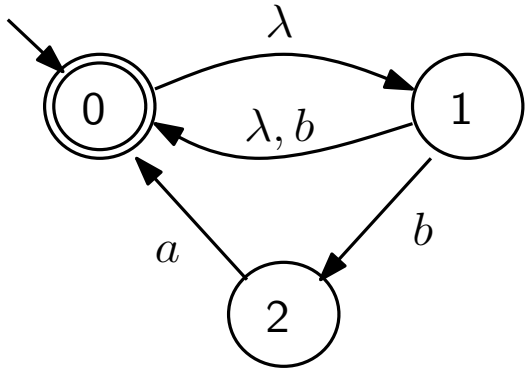


λ-NFAs

A λ-NFA is an NFA that has a λ-transition.

A λ-transition

- allows to move between states without reading the input
- makes the FA highly nondeterministic



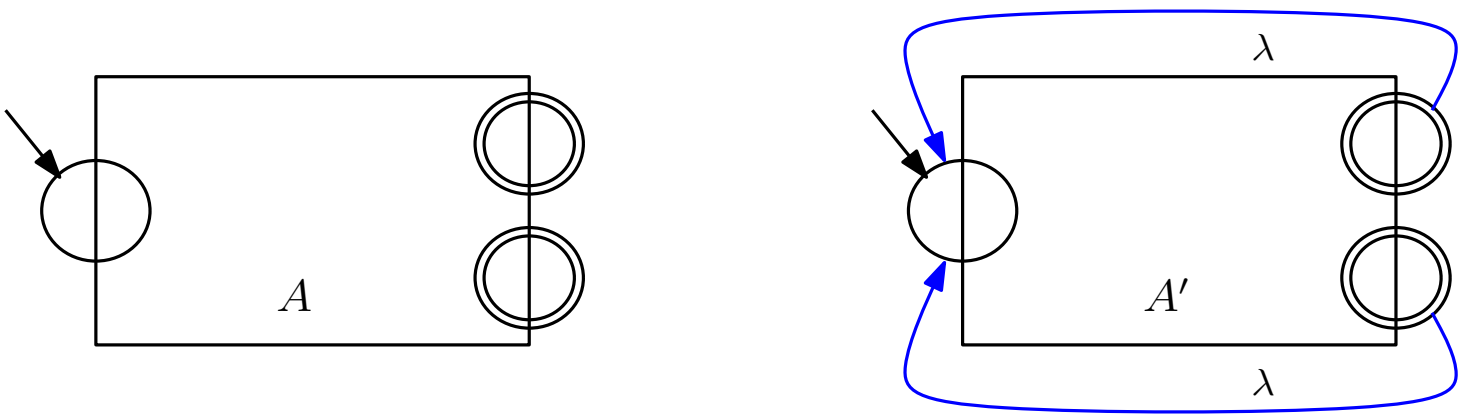
Given an input $baba$

$$(1, baba) \vdash \begin{cases} (0, baba) & \text{by the } \lambda\text{-transition} \\ (0, aba) & \text{by the } b\text{-transition to 0} \\ (2, aba) & \text{by the } b\text{-transition to 2} \end{cases}$$

λ-NFAs

There are situations in whcih λ-transitions are useful.

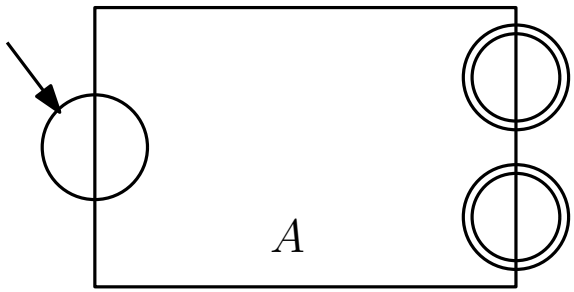
- The plus of an NFA: Given an NFA A, we can feed the final states back to the start state using λ-transitions to give A^+

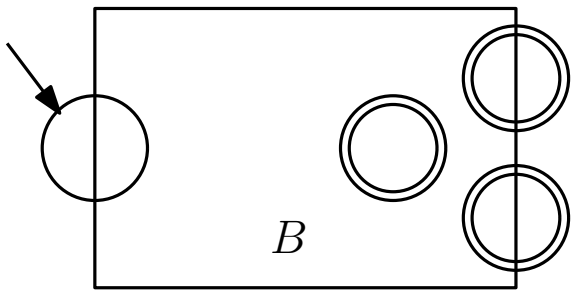


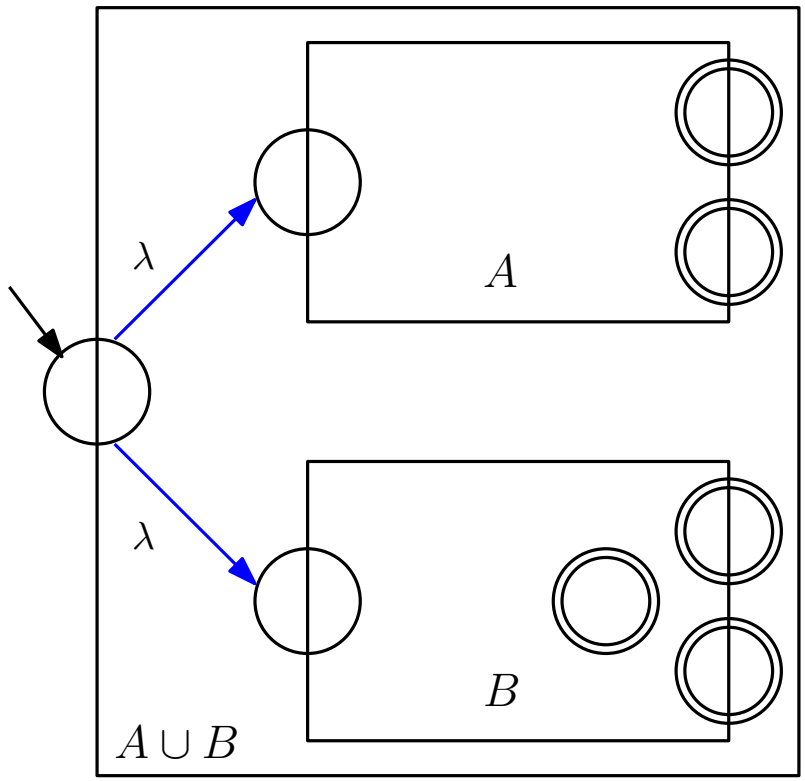
λ-NFAs

There are two situations in whcih λ-transitions are useful.

- The union of two NFAs: Given two NFAs A and B, we can make their union C that consists of A and B together with a new start state which via λ-transitions leads to the start states of A and B.







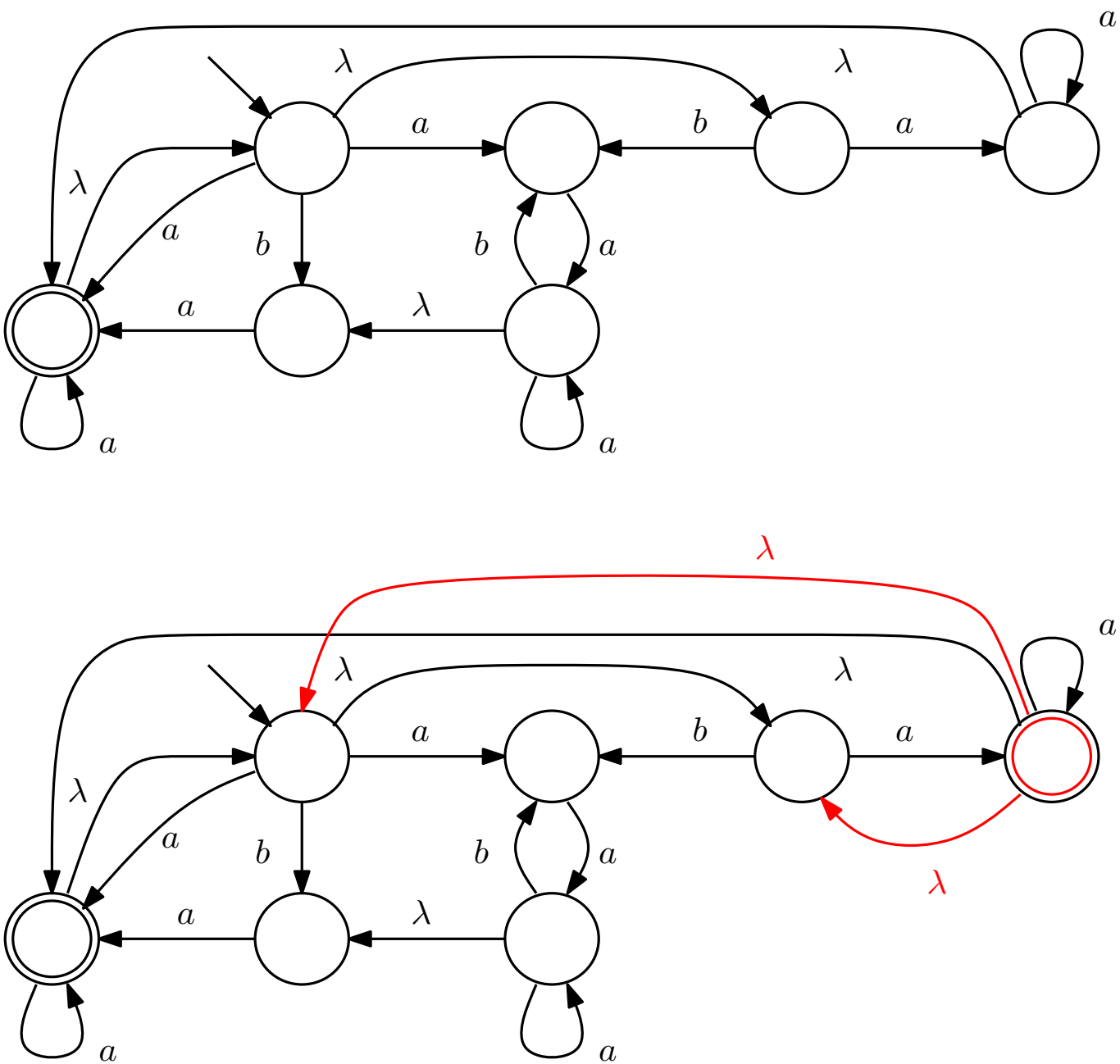
NFAs and λ-NFAs

Every NFA is a λ-NFA but *surprisingly* the expressive power of λ-NFAs is no more than of NFAs. In other words, we can transform a λ-NFA into an equivalent NFA.
Given a λ-NFA $A = (Q, \Sigma, \delta, s, F)$:

1. We modify A so that whenever there is a configuration $(p, w) \vdash^i (q, w)$ for $i \geq 2$, A has a transition (p, λ, q) and whenever a state can reach a final state via λ-transition, we make it to be a final state. Let A'' be the resulting λ-NFA

2. Whenever we have a configuration $(p, \sigma w) \vdash (q, \sigma w) \vdash (r, w)$ in A'', we add a transition (p, σ, r) to A''. Now we do not need to examine the configuration $(p, \sigma w) \vdash^i (q, \sigma w) \vdash (r, w)$ and make use of the new short cut. Finally, we remove all λ-transitions

The proof of correctness is left for an exercise.


a
λ
λ
a
b
a
λ
a
b
b
a
a
λ
a
a
λ
λ
a
λ
λ
a
b
a
λ
a
b
b
a
λ
a
λ
a
a

Summary of Unit

- Finite-state automata (FAs)
- Deterministic finite-state automata (DFAs) and configurations
- Nondeterministic finite-state automata (NFAs) and configurations
- λ-NFAs