A NDFA with λ moves will be denoted **NDFA-λ** and is defined as a 5-tuple

M = (Q, Σ, Δ, S, F)

1. Q is a finite set of states
2. Σ is an alphabet
3. Δ is the nondeterministic state transition function

$$Q \times \{\Sigma \cup \lambda\} \longrightarrow 2^Q$$

4. S is the initial state, S ∈ Q
5. F is the set of final states, F ⊆ Q

λ-CLOSURE

The λ-closure of a set, Z, in an NDFA-λ denoted by machine M is a subset of Q defined using a function

$$\lambda C(Z)$$

that generates a subset of states $\{\lambda C(Z)\} \in Q$.

We can now define λ-closure recursively:

1. Every member of Z is a member of $\lambda C(Z)$
2. For any state $q_j \in \lambda C(Z)$, every state which can be reached by the transition function $\Delta(q_j, \lambda)$ is also an element of $\lambda C(Z)$
3. Nothing else is an element of $\lambda C(Z)$.

The λ-closure of a state, q_i, in an NDFA-λ denoted by machine M is given by

$$\lambda C(\{q_i\})$$

We can relax the notation and just write,

$$\lambda C(q_i)$$

In simple terms, the λ-closure of a state, q_i, includes all the states that are reachable from q_i by λ-moves without scanning any input.

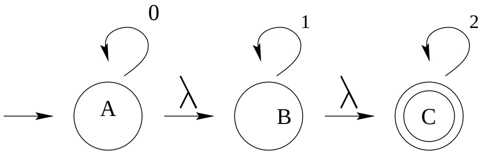


Figure 1: An NDFA with λ moves

$$\lambda C(A) = \{A, B, C\}, \quad \lambda C(B) = \{B, C\}, \quad \lambda C(C) = \{C\}$$

Theorem: Let L $\subseteq \Sigma^*$ be a language recognized by an NDFA with λ moves, **NDFA-λ**, M = (Q, Σ, Δ, S, F). L is also recognized by equivalent DFA.

Proof by construction:

Step 1. We will show by construction that there exits an NDFA (without lambda moves),

$$M^2 = (Q^2, \Sigma^2, \Delta^2, S^2, F^2)$$

that recognizes L.

Step 2. Show that the resulting NDFA is emulated by a DFA. We have already proven Step 2 by construction.

The set of states, the alphabet and
the initial state do not change.

- $Q^2 = Q$
- $\Sigma^2 = \Sigma$
- $S^2 = S$

To define the set of final states, note that every state that can reach a final state by λ moves also functions as a final state.

Thus, foreach state $q_i \in Q$,
if $\lambda C(q_i) \cap F$ is not empty
then $q_i \in F^2$.

The main part of the construction is to build Δ^2 and show that it is correct. To do this, we need to define all the ways of going from a state q_i to a state q_j in NDFA-λ, while scanning exactly one element $x \in \Sigma$ (this includes using lambda moves).

If we do this, we can remove all of the λ moves and thus define Δ^2.

How can we define all the ways of going from a state q_i to a state q_j while scanning exactly one input element?

First, all the states reachable from q_i using λ-moves is given by $\lambda C(q_i)$.

Second, all the ways of scanning input symbol $x \in \Sigma$ from the set of states $\lambda C(q_i)$ will be represented by

$$\Delta(\{\lambda C(q_i)\}, x)$$

Third, after scanning x, we are still free to makes moves using λ transtitions, thus all the ways of going from a state q_i to a state q_j while scanning x is given by

$$\lambda C(\Delta(\{\lambda C(q_i)\}, x)).$$

For each state pair (q_i, q_j), where

$$q_j \in \lambda C(\Delta(\{\lambda C(q_i)\}, x))$$

construct a transition

$$\Delta^2(q_i, x) \longrightarrow q_j.$$

This completes the construction and proof.

Example:

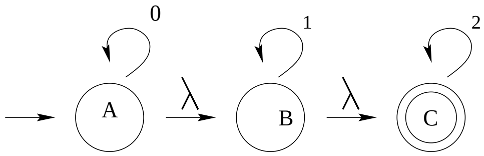


Figure 2: An NDFA with λ moves

Here, $\lambda C(A) = \{A, B, C\}$.

All the ways to scan a 0 from state A is given by

$$\Delta(\{\lambda C(A)\}, 0)$$

$$\Delta(\{A, B, C\}, 0)$$

$$\{A\}$$

And finally, the $\lambda C(\{A\}) = \{A, B, C\}$.

Thus, we construct the following transitions

$$\Delta^2(A, 0) \longrightarrow \{A, B, C\}$$

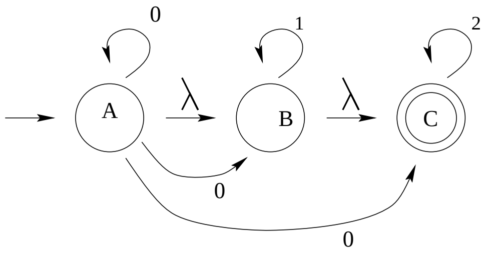


Figure 3: An NDFA with λ moves

We will do the construction in place, then remove the λ transitions.

All the ways to scan a 1 from state A is given by

$$\Delta(\{\lambda C(A)\}, 1)$$

$$\Delta(\{A, B, C\}, 1)$$

$$\{B\}$$

And finally, the $\lambda C(\{B\}) = \{B, C\}$.

Thus, we construct the following transitions

$$\Delta^2(A, 1) \longrightarrow \{B, C\}$$

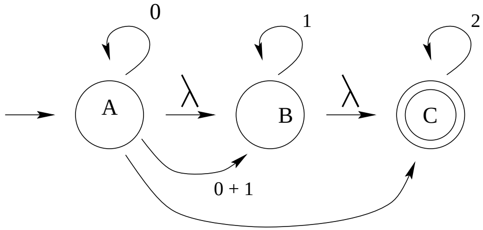


Figure 4: An NDFA with λ moves

All the ways to scan a 2 from state A is given by

$$\Delta(\{\lambda C(A)\}, 2)$$

$$\Delta(\{A, B, C\}, 2)$$

$$\{C\}$$

And the $\lambda C(\{C\}) = \{C\}$.

Thus, we construct the following transition

$$\Delta^2(A, 2) \longrightarrow \{C\}$$

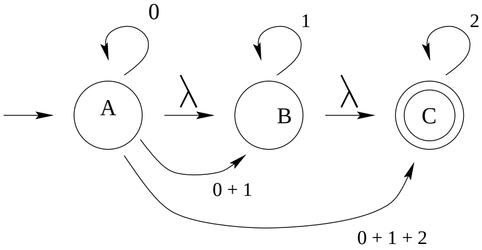


Figure 5: An NDFA with λ moves

For State B

All the ways to scan a 0 from state B is given by

$$\Delta(\{\lambda C(B)\}, 0)$$

$$\Delta(\{B, C\}, 0)$$

$$\emptyset$$

Since the $\lambda C(\emptyset) = \emptyset$.

There are no new transitions.

All the ways to scan a 1 from state B is given by

$$\Delta(\{\lambda C(B)\}, 1)$$

$$\Delta(\{B, C\}, 1)$$

$$\{B\}$$

And the $\lambda C(\{B\}) = \{B, C\}$.

Thus, we construct the following transitions

$$\Delta^2(B, 1) \longrightarrow \{B, C\}$$

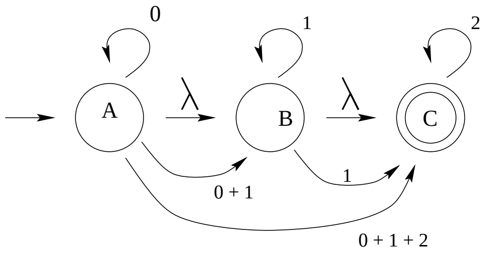


Figure 6: An NDFA with λ moves

All the ways to scan a 2 from state B is given by

$$\Delta(\{\lambda C(B)\}, 2)$$

$$\Delta(\{B, C\}, 2)$$

$$\{C\}$$

And the $\lambda C(\{C\}) = \{C\}$.

Thus, we construct the following transition

$$\Delta^2(B, 2) \longrightarrow \{C\}$$

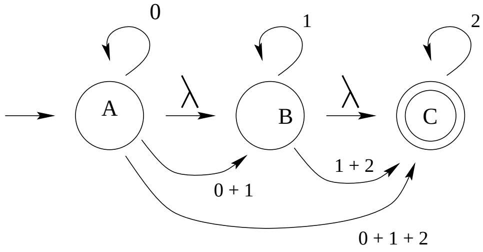


Figure 7: An NDFA with λ moves

For State C

All the ways to scan a 0 from state C is given by

$$\Delta(\{\lambda C(C)\}, 0)$$

$$\Delta(\{C\}, 0)$$

$$\emptyset$$

Since the $\lambda C(\emptyset) = \emptyset$.

There are no new transitions.

All the ways to scan a 1 from state C is given by

$$\Delta(\{\lambda C(C)\}, 1)$$

$$\Delta(\{C\}, 1)$$

$$\emptyset$$

Since the $\lambda C(\emptyset) = \emptyset$.

There are no new transitions.

All the ways to scan a 2 from state C is given by

$$\Delta(\{\lambda C(C)\}, 2)$$

$$\Delta(\{C\}, 2)$$

$$\{C\}$$

And the $\lambda C(\{C\}) = \{C\}$.

Thus, we construct the following transition

$$\Delta^2(C, 2) \longrightarrow \{C\}$$

Note, this is already in the original machine.

We have now added all the new transitions.
Make all states that can reach a final state by λ moves a final state and remove the λ moves from the machine.

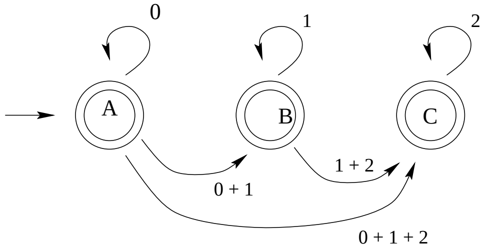


Figure 8: The NDFA without λ moves

A SPECIAL CASE

What happens if we loop on λ moves?

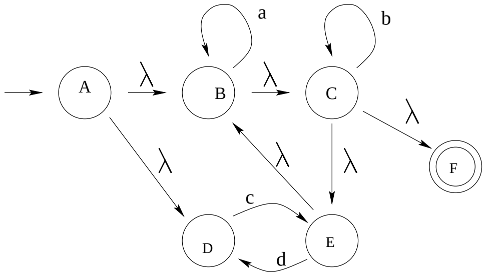


Figure 9: Looping in an NDFA-λ

Note that states B, C and D are connected by directed λ transitions so that one can go to state B, C or D on λ.

One can thus have infinite loops.
(This isn't true in a DFA or NFDA.)

All states that are located on a λ-loop are equivalent in terms of what can be recognized
(i.e., reached, or scanned) from these states.

One can remove λ moves without removing loops, but the machine will be more complicated.

For any state q_i and q_j that are on a λ loop:

$$\lambda C(q_i) = \lambda C(q_j)$$

and thus

$$\lambda C(\Delta(\{\lambda C(q_i)\}, x)) = \lambda C(\Delta(\{\lambda C(q_j)\}, x))$$

This relation is reflexive, symmetric and transitive, which induces an equivalence class.

States that are on a λ loop are computationally equivalent and can therefore be collapsed into one state:

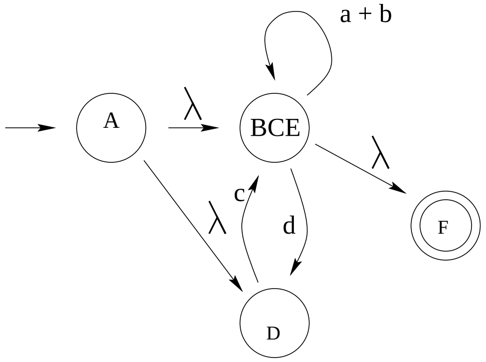
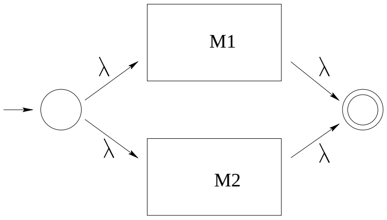


Figure 10: Looping in an NDFA-λ

Theorem: The set of Languages recognized by Finite State Machines are closed under Union.

Proof by Construction using an NDFA-λ:

Assume language $L1$ is recognized by an FSA $M1$ and language $L2$ is recognized by an FSA $M2$. We construct a machine that unions the two machines and hence recognizes $L1 \cup L2$.

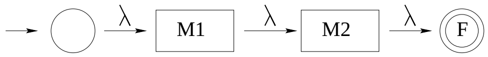


Connect the new initial state to the initial states of M1 and M2.
Connect the new final state to the final states of M1 and M2.

Theorem: The set of Languages recognized by Finite State Machines are closed under Concatenation.

Proof by Construction using an NDFA-λ:

Assume language $L1$ is recognized by an FSA $M1$ and language $L2$ is recognized by an FSA $M2$. We construct a machine that concatenates the two machines and hence recognizes $L1 \cdot L2$.

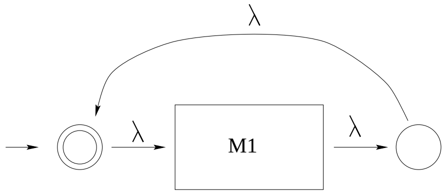


Connect the new initial state to the initial state of M1.
Connect the final states of M1 to the initial state of M2.
Connect the final states of M2 to the new final state.
Make F the only final state.

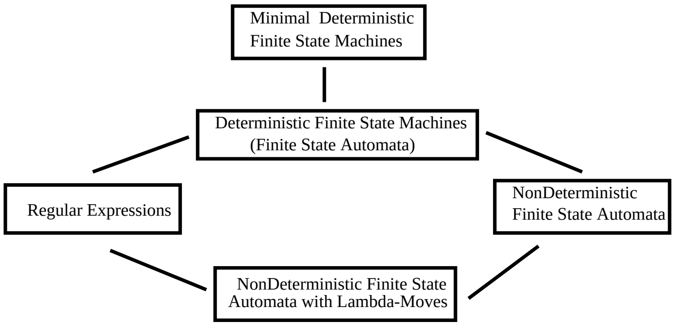
Theorem: The set of Languages recognized by Finite State Machines are closed under Kleene Closure.

Proof by Construction using an NDFA-λ:

Assume language $L1$ is recognized by an FSA $M1$. We construct a machine that computes $(L1)^*$.



Connect the new initial state to the initial state of M1.
Connect the final states of M1 to the new final state.
Connect the new final state to the new initial state.



Minimal Deterministic Finite State Machines

Deterministic Finite State Machines (Finite State Automata)

Regular Expressions

NonDeterministic Finite State Automata

NonDeterministic Finite State Automata with Lambda-Moves