

A **N DFA** is defined as a 5-tuple

$$M = (K, \Sigma, \Delta, S, F)$$

1. K is a finite set of states
2. Σ is an alphabet
3. Δ is the nondeterministic state transition function that maps to some subset of K

$$K \times \Sigma \longrightarrow 2^K$$

2^K is the power set of the set of states

Slide Lecture 4 –80

4. S is the initial state, $S \in K$
(some texts allow multiple initial states . . . S is a set of initial states, $S \subseteq K$)
5. F is the set of final states, $F \subseteq K$

Equivalent automata:

Two finite automata, M_1 and M_2 , are **equivalent** iff

$$L(M_1) = L(M_2)$$

Slide Lecture 4 –81

A NFA has an equivalent nondeterministic regular expression

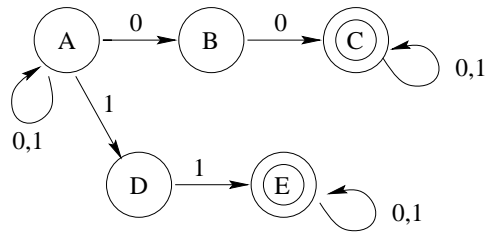


Figure 1: The NFA

Slide Lecture 4 -82

$$A = 0A + 0B + 1A + 1D$$

$$B = 0C$$

$$C = 0C + 1C + \lambda$$

$$D = 1E$$

$$E = 0E + 1E + \lambda$$

Slide Lecture 4 -83

$$A = 0A + 1A + 00C + 11E$$

$$C = (0 + 1)^*$$

$$E = (0 + 1)^*$$

After removing B and D ... and applying Arden's Rule to C and E.

$$A = (0 + 1) A + 00 (0 + 1)^* + 11 (0 + 1)^*$$

OR

$$A = (0 + 1)^* (00 + 11) (0 + 1)^*$$

Slide Lecture 4 -84

The **Theory of Computability** addresses issues concerning what classes of languages can be recognized by various models of automata.

A critical issue for computability: does nondeterminism increase the computing power of finite automata?

For finite state machines: NO. For other machines, maybe.

Computability vs. Complexity

Computability: ability to recognize languages

Complexity: the cost (time and space requirements) of recognizing languages

Slide Lecture 4 -85

Theorem: for each NFA there is an equivalent DFA.

Proof by construction in the form of an algorithm.

Our algorithm takes as input an NFA M , and outputs an equivalent DFA M'

Let $M = (K, \Sigma, \Delta, S, F)$ be a NFA that accepts L

$$M' = (K', \Sigma', \delta, S', F')$$

$$1. K \longrightarrow K'$$

K' will be a subset of 2^K which we define using this analogy: view a NFA as a parallel processor pursuing all possible paths concurrently. Each configuration corresponds to a DFA state.

Slide Lecture 4 –86

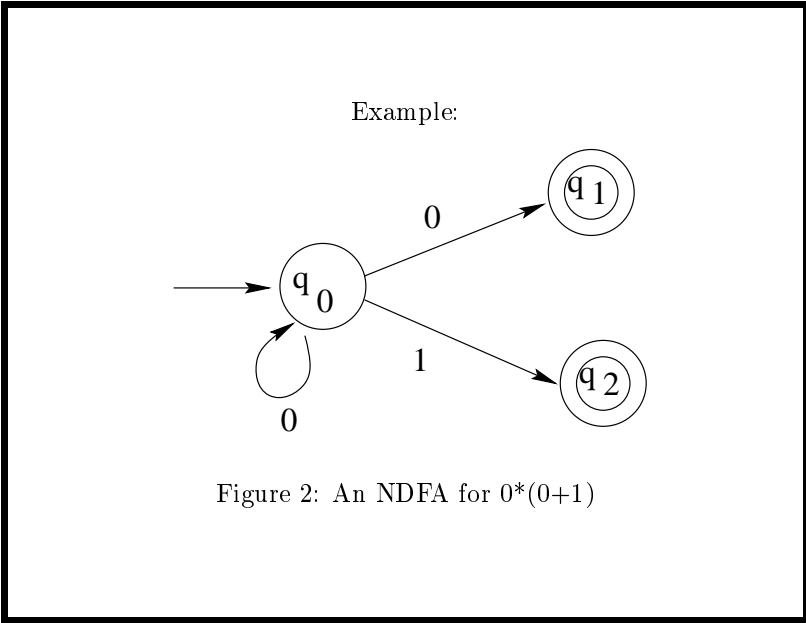
We then create (state, alphabet*) pairs (Q', x) where $Q' \in 2^K$ and $x \in \Sigma^*$ (i.e., a configuration)

Given $\omega = vx$ where $(v \in \Sigma, x \in \Sigma^*)$ as the initial input Q' represents a set

$$Q' = \{q \in K \mid (S, \omega) \stackrel{\vdash}{M} (q, x)\}$$

“states reachable in one move scanning v ”

Slide Lecture 4 –87



Slide Lecture 4 -88

Let $\omega = 0001$

$$(\{q_0\}, 0001) \stackrel{\vdash}{M} (\{q_0, q_1\}, 001)$$

This implies the existence of a composite state

$$\{q_0, q_1\} \in 2^K$$

We are now interested in states reachable from this new composite state.

$$(\{q_0, q_1\}, 001) \stackrel{\vdash}{M} (\{q_0, q_1\}, 01) \stackrel{\vdash}{M} (\{q_0, q_1\}, 1) \stackrel{\vdash}{M} (\{q_2\}, \lambda)$$

and $q_2 \in F$ implies $\omega \in L(M)$

Slide Lecture 4 -89

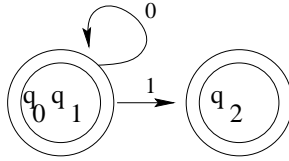


Figure 3: Merging q_0 and q_1

What about $\{q_0, q_1\}$?

If we trace the following computation

$$(\{q_0, q_1\}, 00) \stackrel{\vdash}{M} (\{q_0, q_1\}, 0)$$

we end in $\{q_0, q_1\}$ and accept.

So, if a “composite” state contains an accepting state, the composite state is also an accepting state.

Slide Lecture 4 –90

We therefore define our machine by defining a new set of composite states that are a subset of 2^K (Q_1, \dots, Q_n) such that Q_j is a “reachable” member of 2^K

Q_j is “reachable” if there are transitions

$$S' = Q_1$$

$$Q_1 \longrightarrow Q_2 \text{ when scanning } y \in \Sigma$$

In general, for some reachable state Q_i

$$Q_i \longrightarrow Q_j \text{ when scanning } y \in \Sigma$$

$$F' = \{q'_i \mid Q_i \text{ contains a N DFA state } q, q \in F\}$$

$q' \in$ states in the equivalent DFA

Slide Lecture 4 –91

Recall we assume Q_1 of M' always equals S .

Thus $S' = Q_1 = S$.

Therefore, for each transition

$$\Delta(Q_i, x) \longrightarrow Q_j$$

in the N DFA, M ,

we define $\delta(q'_i, x) \longrightarrow q'_j$ in the DFA, M'

Finally, $\Sigma' = \Sigma$

Having defined $(K', \Sigma', \delta, S', F')$

we have completed the construction.

Slide Lecture 4 -92

It is now simple to show $\omega \in \Sigma^*$,

$$\omega \in L(M) \text{ IFF } \omega \in L(M')$$

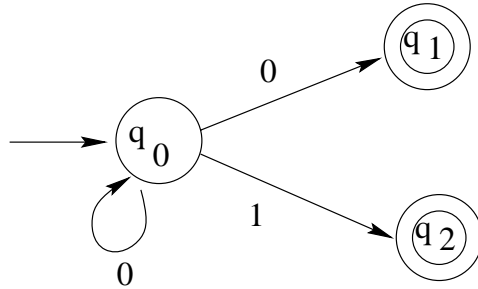
since the DFA emulates all simultaneous parallel transitions of the N DFA.

Recall $Q_1 \dots Q_n$ are some subset of 2^K .

So while N DFA \equiv DFA, N DFA may be more concise!

Slide Lecture 4 -93

Example:



$$K = \{q_0, q_1, q_2\}$$

$$2^K = \{\phi, \{q_0\}, \{q_1\}, \{q_2\}, \\ \{q_0, q_1\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}\}$$

Slide Lecture 4 -94

$$\Sigma = \{0, 1\}$$

$$F = \{q_1, q_2\}$$

$$\Delta = \{ (q_0, 0) \rightarrow q_0 \\ (q_0, 0) \rightarrow q_1 \\ (q_0, 1) \rightarrow q_2 \}$$

Now construct $\Delta(Q_i, x) \Rightarrow Q_j$

$$Q_1 = s = q_0$$

$$\Delta(q_0, 0) \Rightarrow \{q_0, q_1\} \in F'$$

$$\Delta(q_0, 1) \Rightarrow \{q_2\} \in F'$$

Slide Lecture 4 -95

We now know these 3 states are reachable

$$\{q_0\}, \{q_0, q_1\}, \{q_2\}$$

Now determine states reachable from $\{q_0, q_1\}$ and $\{q_2\}$:

$$\Delta (\{q_0, q_1\}, 0) \longrightarrow \{q_0, q_1\} \in F'$$

$$\Delta (\{q_0, q_1\}, 1) \longrightarrow \{q_2\} \in F'$$

$$\Delta (\{q_2\}, 0) \longrightarrow \{\phi\}$$

$$\Delta (\{q_2\}, 1) \longrightarrow \{\phi\}$$

resulting in this new state $\{\phi\} \dots$

What does this do?

$\{\phi\}$ is a trap state.

Slide Lecture 4 -96

Trap states are sometimes left implicit, but to be completely formal

$$\Delta \{ (\phi, 0) \longrightarrow \{\phi\}$$

$$\Delta \{ (\phi, 1) \longrightarrow \{\phi\}$$

So in our DFA M'

$$q'_0 = \{q_0\}$$

$$q'_1 = \{q_0, q_1\}$$

$$q'_2 = \{q_2\}$$

$$q'_3 = \{\phi\}$$

Slide Lecture 4 -97

δ transitions

$$\delta(q'_0, 0) \Rightarrow q'_1$$

$$\delta(q'_0, 1) \Rightarrow q'_2$$

$$\delta(q'_1, 0) \Rightarrow q'_1$$

$$\delta(q'_1, 1) \Rightarrow q'_2$$

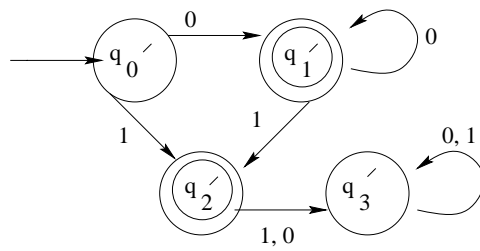
$$\delta(q'_2, 0) \Rightarrow q'_3$$

$$\delta(q'_2, 1) \Rightarrow q'_3$$

$$\delta(q'_3, 0) \Rightarrow q'_3$$

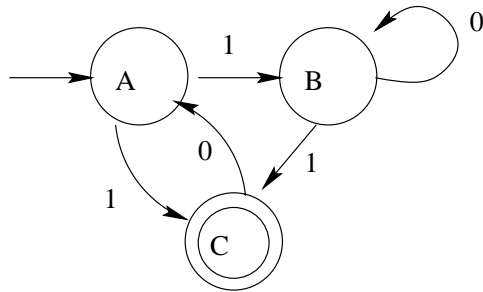
$$\delta(q'_3, 1) \Rightarrow q'_3$$

Slide Lecture 4 -98



Slide Lecture 4 -99

Another example (and more mechanical method)

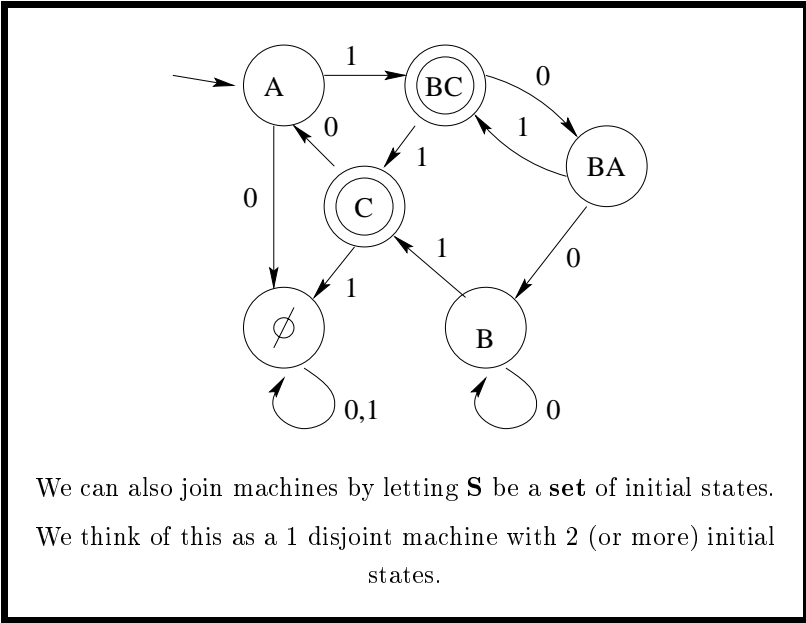


Slide Lecture 4 -100

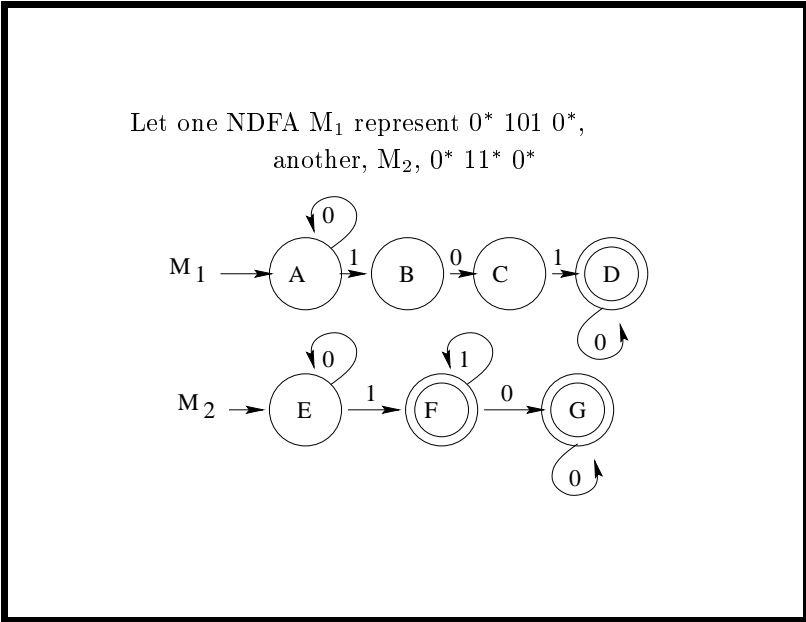
$A \xrightarrow{1} BC \dots \text{new state} \in F'$
 $A \xrightarrow{0} \phi \dots \text{new state}$
 $BC \xrightarrow{1} C \dots \text{new state} \in F'$
 $BC \xrightarrow{0} BA \dots \text{new state}$
 $\phi \xrightarrow{0,1} \phi$
 $C \xrightarrow{1} \phi$
 $C \xrightarrow{0} A$
 $BA \xrightarrow{1} BC$
 $BA \xrightarrow{0} B \dots \text{new state}$
 $B \xrightarrow{0} B$
 $B \xrightarrow{1} C$

No more new reachable states

Slide Lecture 4 -101



Slide Lecture 4 -102



Slide Lecture 4 -103

$S = AE$

$AE \xrightarrow{1} BF \dots \text{new state} \in F'$

$AE \xrightarrow{0} AE$

$BF \xrightarrow{1} F \dots \text{new state} \in F'$

$BF \xrightarrow{0} CG \dots \text{new state} \in F'$

$F \xrightarrow{1} F$

$F \xrightarrow{0} G \dots \text{new state} \in F'$

$CG \xrightarrow{1} D \dots \text{new state} \in F'$

$CG \xrightarrow{0} G$

$G \xrightarrow{1} \phi \dots \text{new state}$

$G \xrightarrow{0} G$

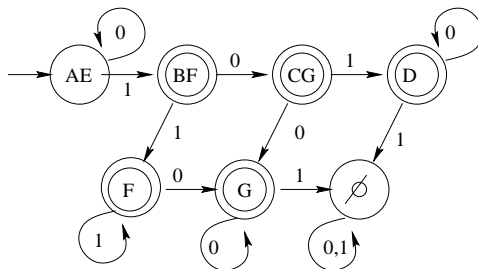
Slide Lecture 4 -104

$D \xrightarrow{1} \phi$

$D \xrightarrow{0} D$

$\phi \xrightarrow{0,1} \phi$

Done.



Slide Lecture 4 -105

A **NFA with lambda-transitions** allows one to change states without reading input. We can think of it as moves on the null string, *lambda*.

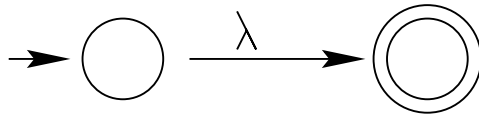


Figure 4: A move on Lambda, λ

Slide Lecture 4 -106

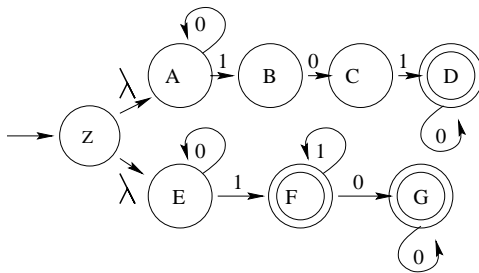


Figure 5: Using lambda-moves to union two machines.

Slide Lecture 4 -107