Automata and Languages

Prof. Mohamed Hamada

Software Engineering Lab.
The University of Aizu
Japan

Regular Expressions (RE)

Empty set	**Φ**	A RE denotes the empty set
Empty string	**λ**	A RE denotes the set {λ}
Symbol	**a**	A RE denotes the set {a}
Alternation	**M + N**	If M is a RE for the set *M* and N is a RE for the set *N*, then M+N is a RE for the set *M* U *N*
Concatenation	**M • N**	If M is a RE for the set *M* and N is a RE for the set *N*, then M.N is a RE for the set *M* . *N*
Kleene-*	**M***	If M is a RE for the set *M*, then M* is a RE for the set *M**

Regular Expressions (RE)

Operation	Notation	Language	UNIX
Alternation	r_1+r_2	$L(r_1)\cup L(r_2)$	$r_1\|r_2$
Concatenation	$r_1 \bullet r_2$	$L(r_1)\bullet L(r_2)$	$(r_1)(r_2)$
Kleene-*	$r*$	$L(r)*$	$(r)*$
Kleene-+	r^+	$L(r)^+$	$(r)+$
Exponentiation	r^n	$L(r)^n$	$(r)\{n\}$

Regular Expressions (RE)

Example

For the alphabet Σ={0, 1}

0+1 is a RE denote the set {0} U {1}

0* is a RE denote the set {0}*={**λ,0,00,…**}

0.1* is a RE denote the set {0}.{**λ**,1,11,…}
={0, 01, 011, …}

Regular Expressions (RE)

Notes

For a RE r, r^i = r.r….r *i*-times
Operations precedence: * > . > +   So we can omit many parentheses, for example: the RE ((0(1*))+0) can be written as 01*+0
We may abbreviate rr* to r^+
The corresponding set (language) denoted by a RE r will be expressed as L(r)

Nondeterministic Finite Automata (NFA)

Definition

A nondeterministic finite automaton (NFA) M is defined by a 5-tuple $M=(Q,\Sigma,\delta,q_0,F)$, with

- Q: finite set of states
- Σ: finite input alphabet
- δ: transition function $\delta: Q\times\Sigma \rightarrow P(Q)$
- $q_0 \in Q$: start state
- $F \subseteq Q$: set of final states

Nondeterministic Finite Automata (NFA)

Definition

A string *w* is ***accepted*** by an NFA *M* if and only if *there exists* a path starting at q_0 which is labeled by *w* and ends in a final state.

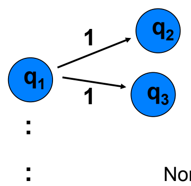
The ***language accepted by*** an NFA *M* is the set of all strings which are accepted by *M* and is denoted by *L* (*M*).

$$L(M)=\{w: \delta(q_0,w) \cap F \neq \Phi\}$$

Nondeterministic Finite Automata (NFA)

Definition

A nondeterministic finite automaton has transition rules like:



Nondeterministic transition

Nondeterministic Finite Automata (NFA)

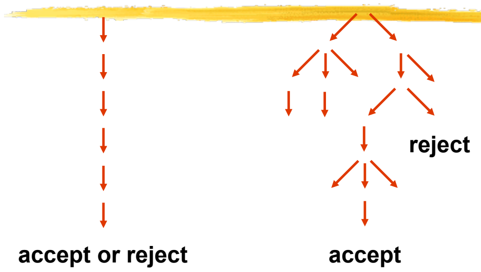
Nondeterminism ~ Parallelism

For any string w, the nondeterministic automaton can be in a subset $\subseteq$ Q of several possible states.

If the final set contains a final state, then the automaton accepts the string.

"The automaton processes the input in a parallel fashion; its computational path is no longer a line, but more like a tree".

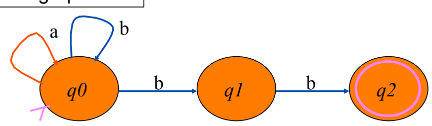
Deterministic Computation | **Non-Deterministic Computation**



Nondeterministic Finite Automata (NFA)

We can write the NFA in two ways

1. State digraph



2. Table

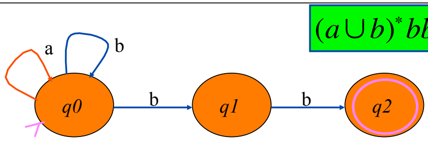
d	a	b
q0	{q0}	{q0,q1}
q1	f	{q2}
q2	f	f

Nondeterministic Finite Automata (NFA)

Example 1

Write an NFA for the language, over Σ={a,b}, ending in bb

$(a \cup b)^* bb$



$Q = \{q0, q1, q2\}$

$\Sigma = \{a, b\}$

$F = \{q2\}$

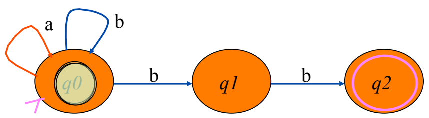
Check the input abb?

Nondeterministic Finite Automata (NFA)

Quiz

Check the input abb?

$(a \cup b)^* bb$



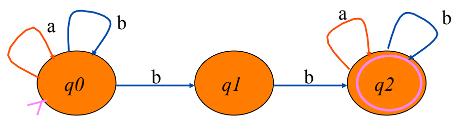
Input: | a | b | b |

q2 is a final state hence the input abb is accepted

Nondeterministic Finite Automata (NFA)

Example 2

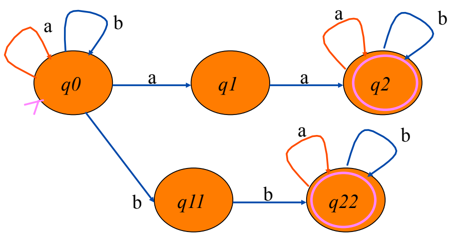
Write an NFA for the language, over Σ={*a*,*b*}, L=(*a* ∪ *b*)* *bb* (*a* ∪ *b*)*



Nondeterministic Finite Automata (NFA)

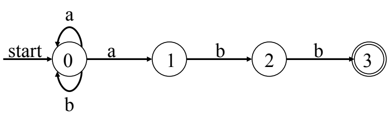
Example 3

Write an NFA for the language, over Σ={*a*,*b*}, L=(*a* ∪ *b*)* (*aa* ∪ *bb*) (*a* ∪ *b*)*



Nondeterministic Finite Automata (NFA)

Example 4



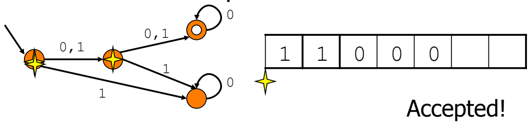
What language is accepted by this NFA?

Answer: (a+b)*abb

Nondeterministic Finite Automata (NFA)

Example 5

For example, consider the following NFA which reads the input 11000.



Accepted!

NFA → DFA

Theorem: For every language L that is accepted by a nondeterministic finite automaton, there is a (deterministic) finite automaton that accepts L as well. DFA and NFA are equivalent computational models.

Proof idea: When keeping track of a nondeterministic computation of an NFA N we use many 'fingers' to point at the subset ⊆ Q of states of N that can be reached on a given input string.
We can simulate this computation with a deterministic automaton M with state space P(Q).

NFA → DFA

Proof

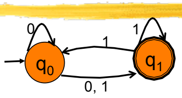
Let L be the language recognized by the NFA N = (Q,Σ,δ,q_0,F). Define the DFA M = $(Q',\Sigma,\delta',q'_0,F')$ by

1. $Q' = P(Q)$
2. $\delta'(R,a) = \{ q \in Q \mid q \in \delta(r,a) \text{ for an } r \in R \}$
3. $q'_0 = \{q_0\}$
4. $F' = \{R \in Q' \mid R$ contains a 'final state' of N$\}$

It is easy to see that the previously described deterministic finite automaton M accepts the same language as N.

NFA → DFA

Example 1

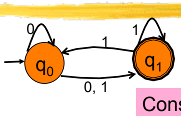
Convert the NFA:  into a DFA?

Given NFA		Constructed DFA
$Q=\{q_0, q_1\}$	⟹	$Q'=P(Q)=\{\Phi, \{q_0\}, \{q_1\}, \{q_0, q_1\}\}$
q_0	⟹	$q'_0 = \{q_0\}$
$F=\{q_1\}$	⟹	$F'=\{\{q_1\}, \{q_0, q_1\}\}$

For δ' see the next slide

NFA → DFA

Example 1

Convert the NFA:  into a DFA?

Given NFA

$\delta(q_0,0)=\{q_0,q_1\}$ ⟹ $\delta'(\{q_0\},0)=\{q_0,q_1\}$

$\delta(q_0,1)=\{q_1\}$ ⟹ $\delta'(\{q_0\},1)=\{q_1\}$

$\delta(q_1,0)=\Phi$ ⟹ $\delta'(\{q_1\},0)=\Phi$

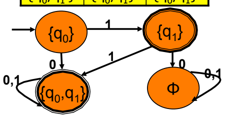
$\delta(q_1,1)=\{q_0,q_1\}$ ⟹ $\delta'(\{q_1\},1)=\{q_0,q_1\}$

$\delta'(\{q_0,q_1\},0)=\delta(q_0,0) \cup \delta(q_1,0)=\{q_0,q_1\}$

$\delta'(\{q_0,q_1\},1)=\delta(q_0,1) \cup \delta(q_1,1)=\{q_0,q_1\}$

Constructed DFA

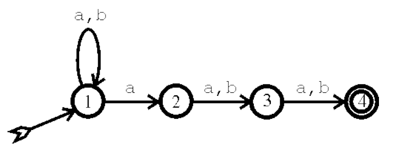
δ'	0	1
Φ	Φ	Φ
$\{q_0\}$	$\{q_0,q_1\}$	$\{q_1\}$
$\{q_1\}$	Φ	$\{q_0,q_1\}$
$\{q_0,q_1\}$	$\{q_0,q_1\}$	$\{q_0,q_1\}$



NFA → DFA

Example 2

Start with the NFA:



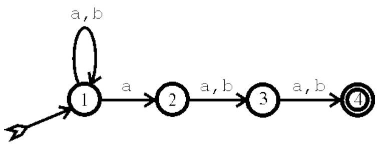
Q1: What's the accepted language?

Q2: How many states does the subset construction create in this case?

NFA → DFA

Example 2

A1: $L = \{x \in \{a,b\}^* \mid 3^{rd}$ bit of x from right is a$\}$



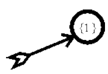
A2: $16 = 2^4$ states.

That's a lot of states. Would be nice if only had to construct useful states, I.e. those that can be reached from start state.

NFA → DFA

Example 2

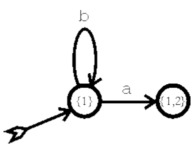
Start with {1}:



NFA → DFA

Example 2

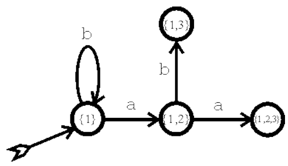
Branch out. Notice that δ(1,a) = {1,2}.



NFA → DFA

Example 2

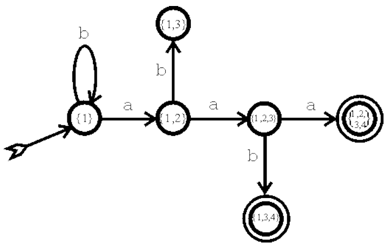
Branch out. Notice that δ' ({1,2},a) = {1,2,3}.



NFA → DFA

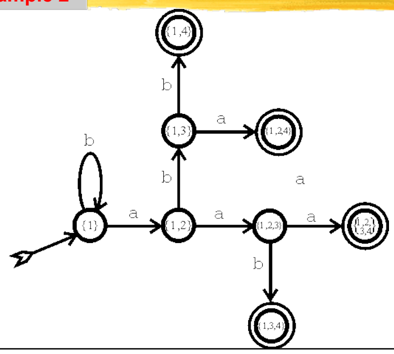
Example 2

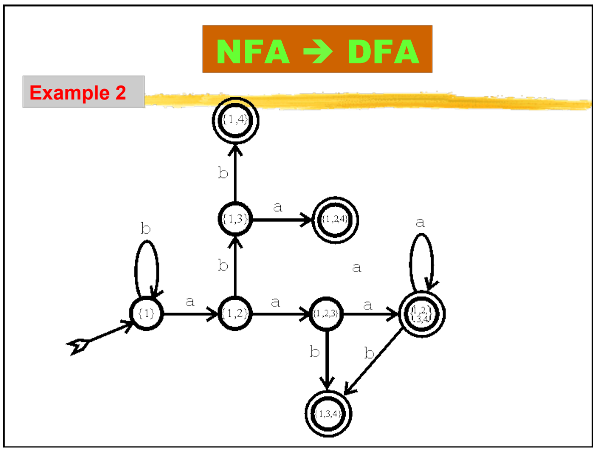
Branch out. Note that δ' ({1,2,3},a) = {1,2,3,4}

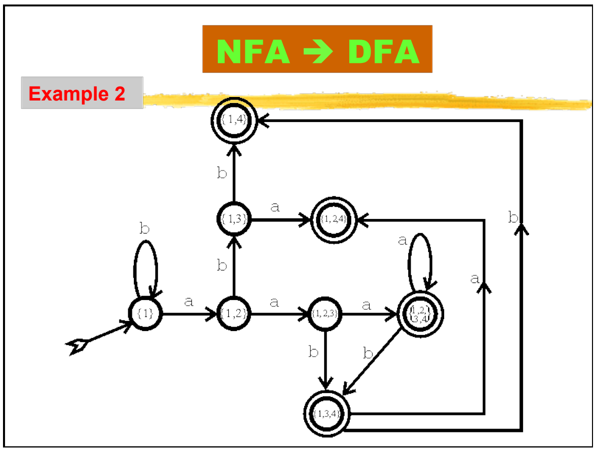


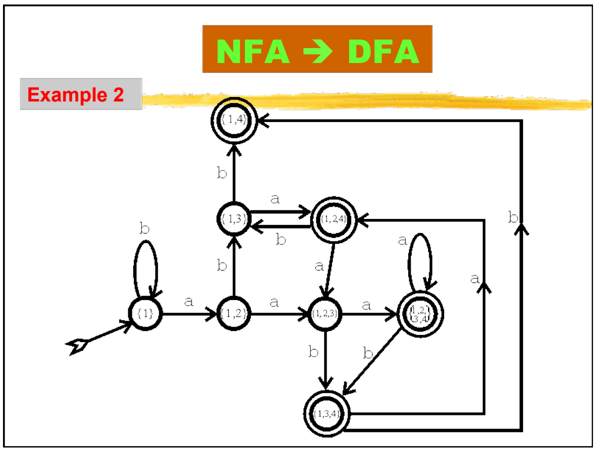
NFA = DFA

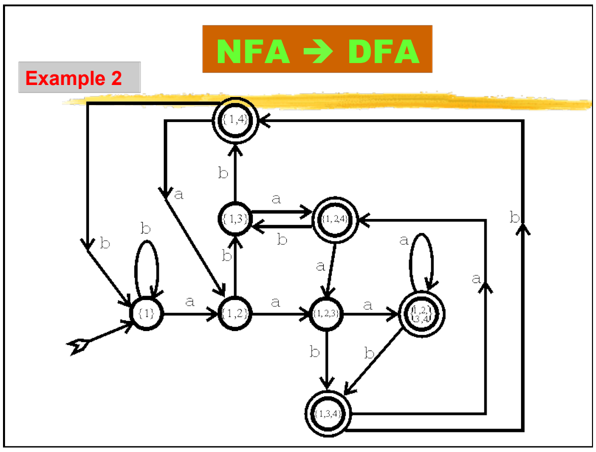
Example 2




NFA → DFA
Example 2


NFA → DFA
Example 2

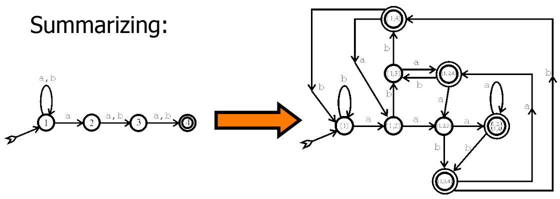

NFA → DFA
Example 2


NFA → DFA
Example 2

NFA → DFA

Example 2

Summarizing:



Therefore, we saved 50% effort by not constructing all possible states unthinkingly.

NFA → DFA

Exercise

Convert the following NFA into an equivalent DFA?