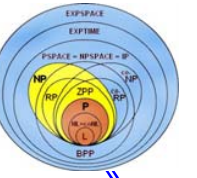
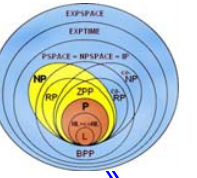
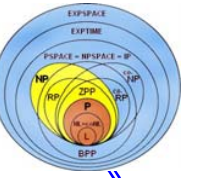


Session 2

- Deterministic Finite Automate
- Nondeterministic Finite Automate
- The equivalence of DFA and NFA



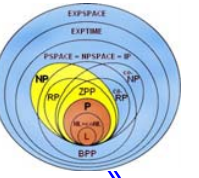
Deterministic Finite Automata



Definition of DFA

A deterministic finite automaton is a quintuple $A = (Q, \Sigma, \delta, q_0, F)$.

- Q is a finite set of *states*
- Σ is a finite set of *input symbols*, or a alphabet
- δ is a *transition function* from $Q \times \Sigma$ to Q: $(q, a) \to p \in Q$
- q_0 is a *start state*, one of the state in Q
- F is a set of *final* or *accepting* states, one of subset of Q



Example Let $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{0, 1\}$, q_0 is the start state, and $F = \{q_1\}$. If transition function δ: $Q \times \Sigma \to Q$ is defined by

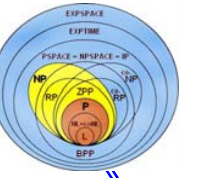
$$\delta(q_0, 0) = q_2, \ \delta(q_1, 0) = q_1, \ \delta(q_2, 0) = q_2,$$

$$\delta(q_0, 1) = q_0, \ \delta(q_1, 1) = q_1, \ \delta(q_2, 1) = q_1,$$

then

$$A_1 = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\})$$

is a DFA.

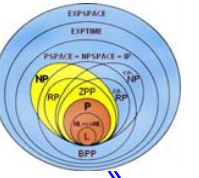


Simple Notations for DFA

Specifying a DFA as a five-tuple with a detailed description of the δ transition function is both tedious and hard to read. There are two preferred notations for describing automata:

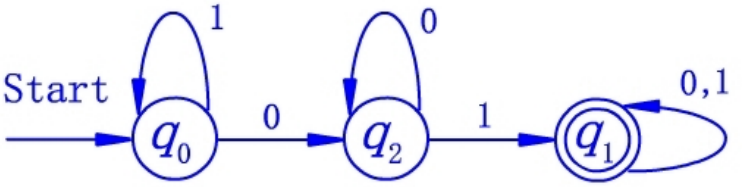
Transition diagram a graph in which the nodes are the states, and arcs are labeled by input symbols, indicating the transitions of that automaton.

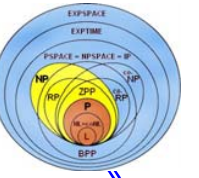
Transition table a tabular listing of the δ function, which by implication tells us the set of states and the input alphabet.



Transition Diagram

The automation $A_1 = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\})$ as a transition diagram

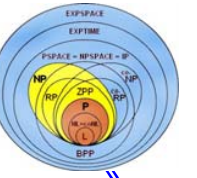




Transition Table

The automation $A_1 = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\})$ as a transition table

	0	1
$\rightarrow q_0$	q_2	q_0
$\star q_1$	q_1	q_1
q_2	q_2	q_1



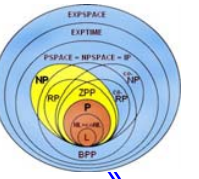
Extended Transition Function

The transition function δ can be extended to $\hat{\delta}$ that operates on states and strings (as opposed to states and symbols) by induction on the length of the input string:

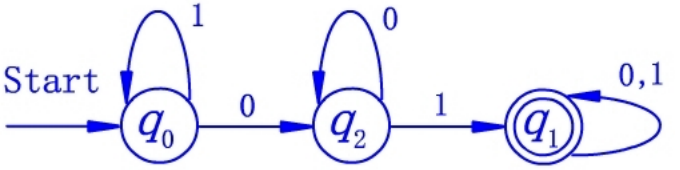
Basis step: $\hat{\delta}(q, \epsilon) = q$.

Inductive step: Suppose $w = xa$, then

$$\hat{\delta}(q, w) = \delta(\hat{\delta}(q, x), a).$$

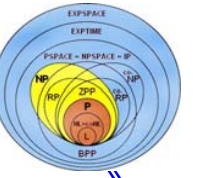


Example For $A_1 = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\})$, we have $\hat{\delta}(q_0, 1101) = q_1$.



In fact,

- $\hat{\delta}(q_0, 1) = \delta(\hat{\delta}(q_0, \epsilon), 1) = \delta(q_0, 1) = q_0$.
- $\hat{\delta}(q_0, 11) = \delta(\hat{\delta}(q_0, 1), 1) = \delta(q_0, 1) = q_0$.
- $\hat{\delta}(q_0, 110) = \delta(\hat{\delta}(q_0, 11), 0) = \delta(q_0, 0) = q_2$.
- $\hat{\delta}(q_0, 1101) = \delta(\hat{\delta}(q_0, 110), 1) = \delta(q_2, 1) = q_1$.



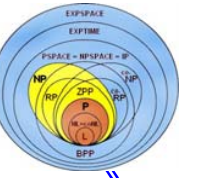
☞ Obviously, for any state q, and input symbol a,

$$\hat{\delta}(q, a) = \delta(q, a).$$

Furthermore, for any strings x and y, one can prove:

$$\hat{\delta}(q, ax) = \hat{\delta}(\delta(q, a), x), \qquad \hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y).$$

Do them!



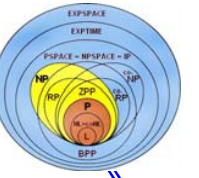
Language of DFA

The language of a DFA $A = (Q, \Sigma, \delta, q_0, F)$ is defined by

$$L(A) = \{w | \hat{\delta}(q_0, w) \in F\}$$

☞ Note that we require that δ, and consequently $\hat{\delta}$, be *total functions*. At each step, a unique move is defined, so that we justified in calling such an automaton deterministic.

A DFA will process *every* string in Σ^* and either accept it or not accept it.

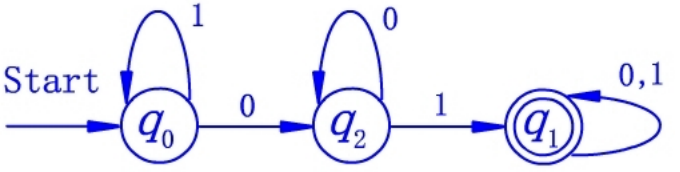


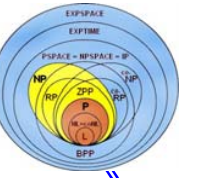
If L is a language for some DFA A, then we say L is a regular language.

To show any language is regular, all we have to do is find a DFA for it.

Example For $\Sigma = \{0, 1\}$, let L is the set consisting of all strings with at least one 01. Show L is regular.

Proof The construction of the DFA for this language is not difficulty. The solution is just $A_1 = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\})$.

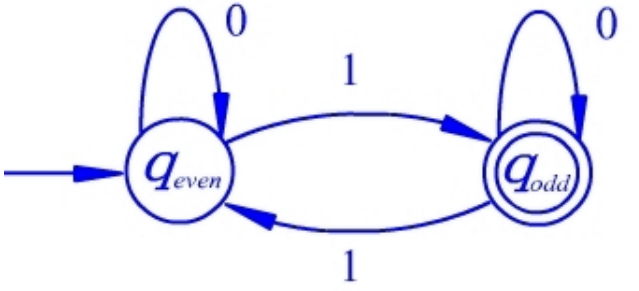


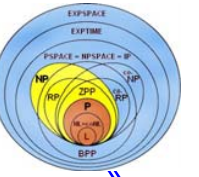


Designing DFA

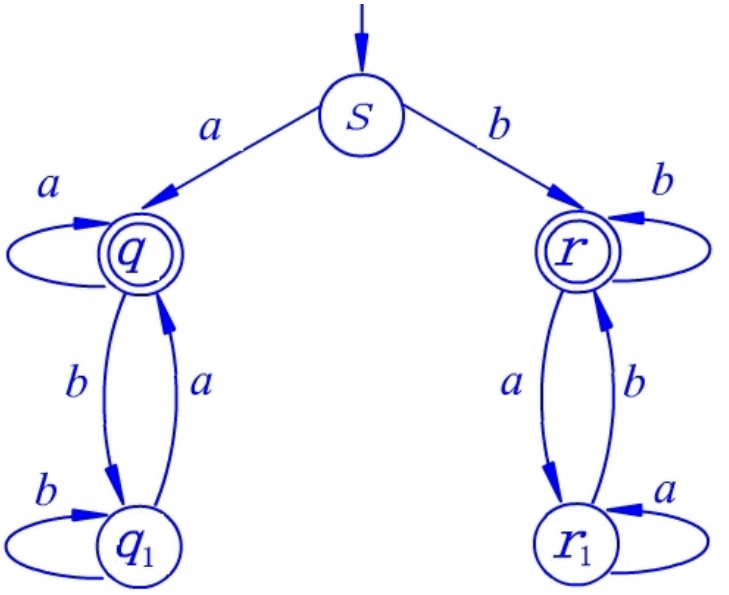
Whether it be of automation or artwork, design is a *creative* process. However, you might find a particular approach helpful when designing various types of automata.

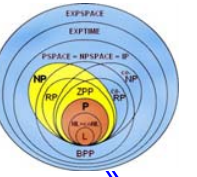
For example, suppose $\Sigma = \{0, 1\}$, design a DFA to accept all strings with an odd number of 1's.



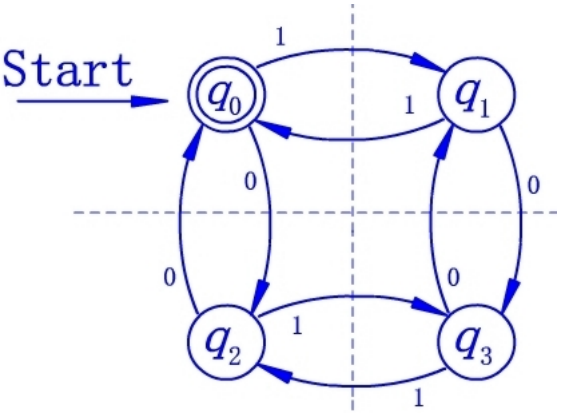


Example Suppose $\Sigma = \{a, b\}$, design a DFA to accept all strings that start and end with a, or start and end with b.

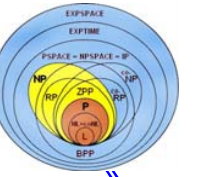




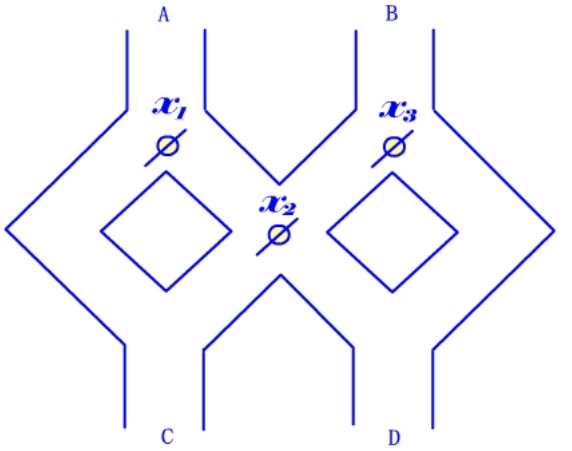
Example Suppose $\Sigma = \{0, 1\}$, design a DFA to accept all and only strings with an even number of 0's and an even number of 1's.

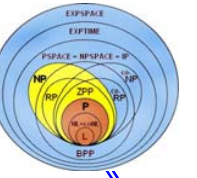


Question How to design a DFA such that to accept all and only with an even number of 0's and a k-multiple number of 1's $(k = 2, 3, \cdots)$.



Example *Marble-rolling toy*. A marble is dropped at A and B. Levers x_1, x_2 and x_3 cause the marble to fall either to the left or to the right. Whenever a marble encounters a lever, it causes the lever to reverse after the marble passes, so the next marble will take the opposite branch.



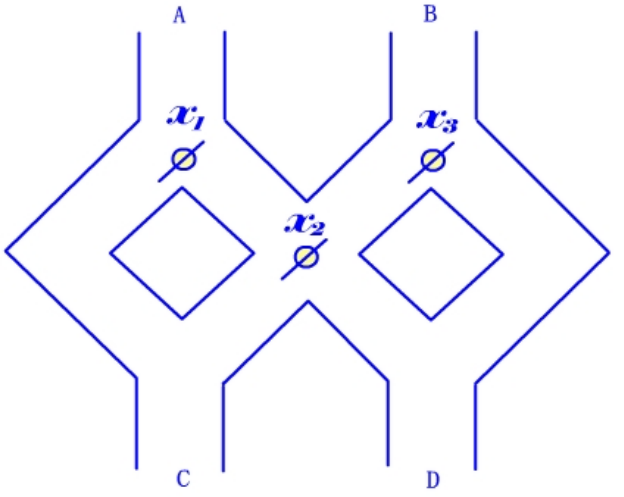


Model this toy by a DFA.

Let the inputs A and B represent the input into which the marble is dropped. Let acceptance correspond to the marble exiting at D; nonacceptance represents a marble exiting at C.

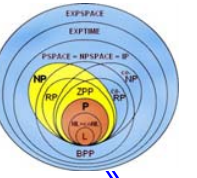
A state is represented as sequence of three bits followed by r or a (previous input *rejected* or *accepted*). For instance, 010a means $left$, $right$, $left$, $accepted$.

Tabular representation of DFA for the toy is given next slide.

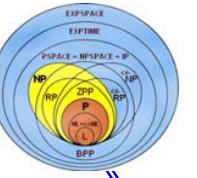


	A	B
$\rightarrow 000r$	$100r$	$011r$
$\star 000a$	$100r$	$011r$
$\star 001a$	$101r$	$000a$
$010r$	$110r$	$001a$
$\star 010a$	$110r$	$001a$
$011r$	$111r$	$010a$
$100r$	$010r$	$111r$
$\star 100a$	$010r$	$111r$
$101r$	$011r$	$100a$
$\star 101a$	$011r$	$100a$
$110r$	$000a$	$101a$
$\star 110a$	$000a$	$101a$
$111r$	$001a$	$110a$

Question Informally describe the language of the automation.



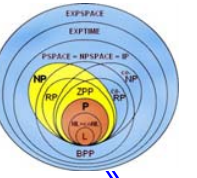
Nondeterministic Finite Automate

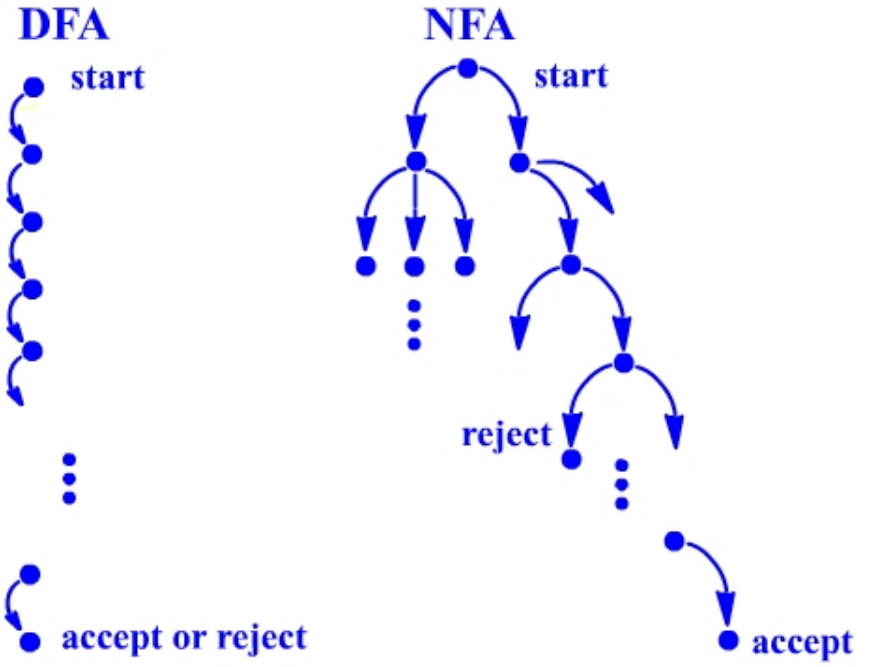


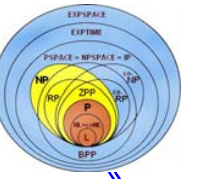
An Informal View of NFA

The difference between the DFA and the NFA is in the type of transition function. For the NFA, transition function takes a state and input symbol as arguments, but returns a set of zero, one, or more states.

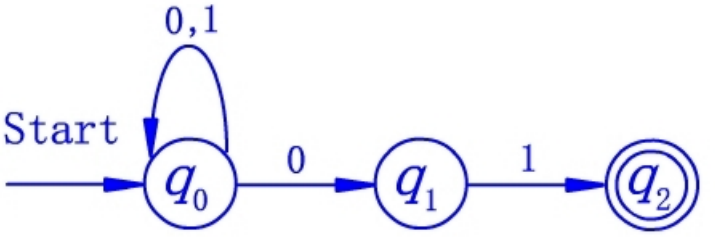
That means, an NFA can be in several states at once, or, viewed another way, it can "guess" which state to go to next.


EXPSPACE
EXPTIME
PSPACE = NPSPACE = IP
NP
coNP
RP
ZPP
coRP
P
L
BPP

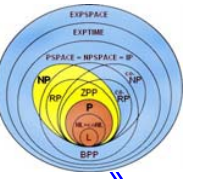

DFA
start
accept or reject
NFA
start
reject
accept



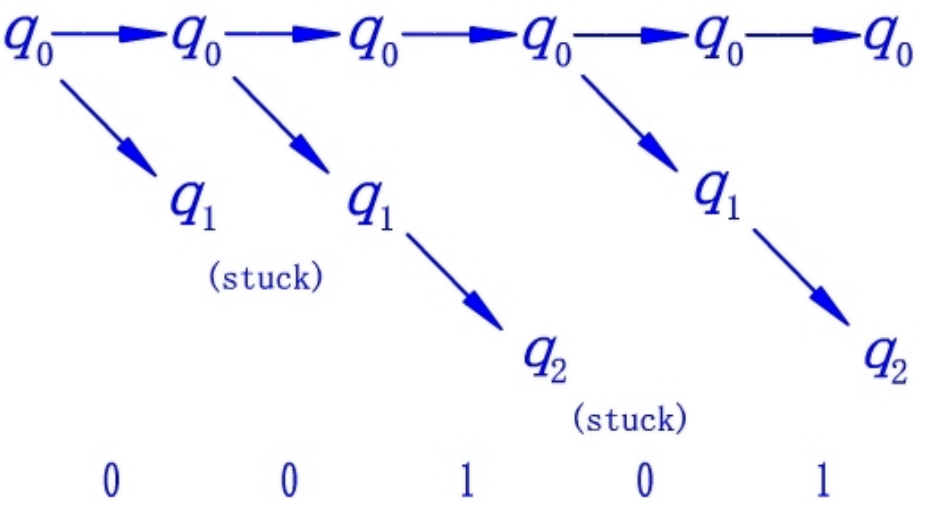
Example An NFA that accepts all and only strings ending in 01.

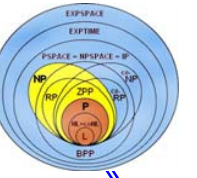


The automation is in q_0. It is possible the the next symbol does not begin the final 01, even if that symbol is 0. Thus, q_0 may transition to its self on both 0 and 1. However, if the next symbol is 0, it guesses that final 01 has begun. It has the option of going either to q_0 or to q_1, and in fact it does both. In q_1, it checks that the next symbol is 1, and if so, it goes to q_2 and accepts.



Here is what happens when the NFA processes the input 00101.

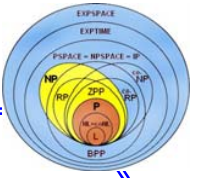




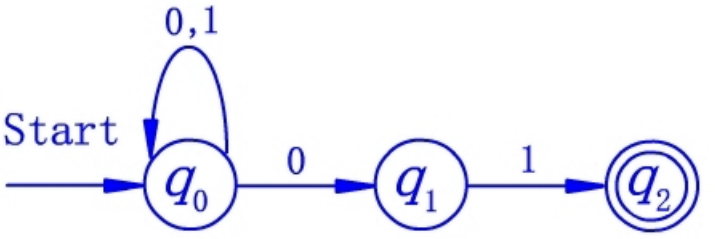
Definition of NFA

A nondeterministic finite automate is a quintuple $A = (Q, \Sigma, \delta, q_0, F)$.

- Q is a finite set of *states*
- Σ is a finite set of *input symbols*, or a alphabet
- δ is a *transition function* from $Q \times \Sigma$ to the powerset of Q: $(q, a) \rightarrow S \subseteq Q$
- q_0 is a *start state*, a member of Q
- F is a set of *final* or *accepting* states, a subset of Q

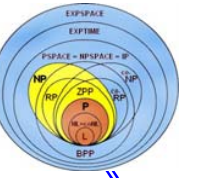


Example The NFA



can be specified formally as $A_2 = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\})$, where δ is the transition function

	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
$\star q_2$	$\emptyset$	$\emptyset$



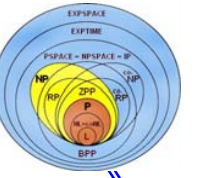
Extended Transition Function

The transition function δ of an NFA can be extended to $\hat{\delta}$:

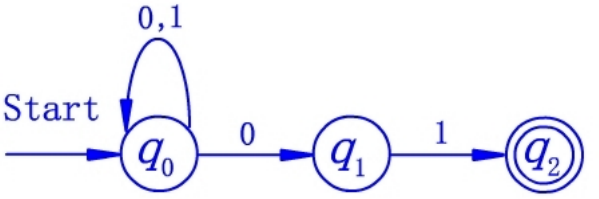
Basis step: $\hat{\delta}(q, \epsilon) = \{q\}$.

Inductive step: Suppose $w = xa$, then

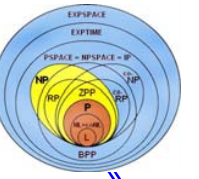
$$\hat{\delta}(q, w) = \bigcup_{p \in \hat{\delta}(q,x)} \delta(p, a).$$



Example Use $\hat{\delta}$ to describe the processing of input 00101 by the NFA A_2



- $\hat{\delta}(q_0, 0) = \delta(q_0, 0) = \{q_0, q_1\}$.
- $\hat{\delta}(q_0, 00) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$.
- $\hat{\delta}(q_0, 001) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$.
- $\hat{\delta}(q_0, 0010) = \delta(q_0, 0) \cup \delta(q_2, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$.
- $\hat{\delta}(q_0, 00101) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$.



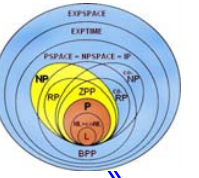
☞ For any state q, and input symbol a,

$$\hat{\delta}(q,a) = \bigcup_{p \in \hat{\delta}(q,\epsilon)} \delta(p,a) = \delta(q,a).$$

Furthermore, for all $q \in Q$ and all $x, y \in \Sigma^*$, one can prove:

$$\hat{\delta}(q,xy) = \bigcup_{p \in \hat{\delta}(q,x)} \hat{\delta}(p,y).$$

Do them!



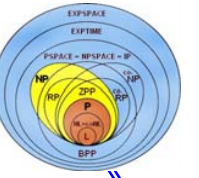
Language of NFA

The language of an NFA $A = (Q, \Sigma, \delta, q_0, F)$ is defined by

$$L(A) = \{w | \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$$

Example For $A_2 = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\})$, we will prove

$$L(A_2) = \{w | w \text{ ends in } 01\}.$$

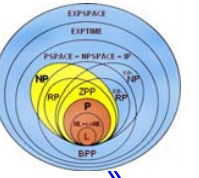


Proof We'll do a mutual induction on the three statements below

1. $\forall w \in \Sigma^* \Rightarrow q_0 \in \hat{\delta}(q_0, w)$
2. $q_1 \in \hat{\delta}(q_0, w) \Leftrightarrow w = x0$
3. $q_2 \in \hat{\delta}(q_0, w) \Leftrightarrow w = x01$

The proof is an induction on w, the length of w, starting with length 0.

Basis step: If $|w| = 0$, then $w = \epsilon$. Then statement (1) follows from the definition. For statements (2) and (3) both sides are false for ϵ.

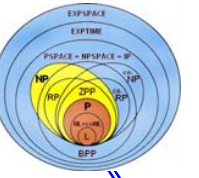


Inductive step: Assume $w = xa$, where $a \in \{0, 1\}$, $|x| = n$ and statements (1), (2) and (3) hold for x. We will show that the statements hold for w.

1. We know that $q_0 \in \hat{\delta}(q_0, x)$. Since there are transitions on both 0 and 1 from q_0 to itself, it follows that $q_0 \in \hat{\delta}(q_0, w)$, so statement (1) is proved for w.

2. (if) Assume that $w = x0$. By statement (1) applied to x, we know that $q_0 \in \hat{\delta}(q_0, x)$. Since there is a transition from q_0 to q_1 on input 0, so $q_1 \in \hat{\delta}(q_0, w)$.

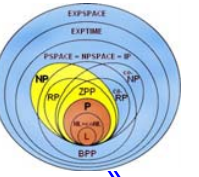
   (only-if) Suppose $q_1 \in \hat{\delta}(q_0, w)$. From the diagram of A_2, we see that the only way to get into state q_1 is if the input sequence w is of the form $x0$.



3. (if) Assume that w ends in 01. Then if $w = x1$, we know that x ends in 0. By statement (2) applied to x, we know that $q_1 \in \hat{\delta}(q_0, x)$. Since there is a transition from q_1 to q_2 on input 1, so $q_2 \in \hat{\delta}(q_0, w)$.

   (only-if) Suppose $q_2 \in \hat{\delta}(q_0, w)$. From the diagram of A_2, we discover that the only way to get into state q_2 is for w to be of the form $x1$, where $q_1 \in \hat{\delta}(q_0, x)$. By statement (2) applied to x, we know that x ends in 0. Thus, w ends in 01, and we have proved statement (3).

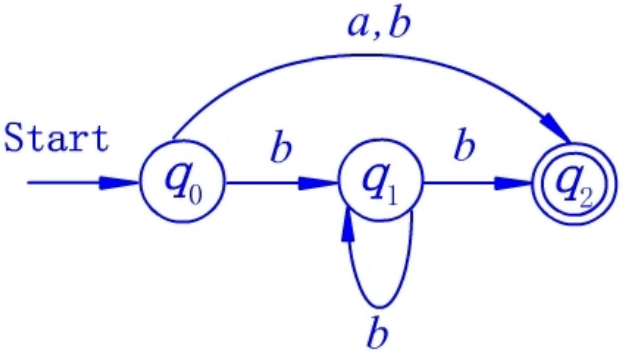
◀

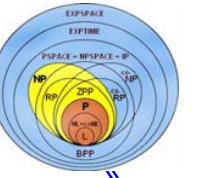


Designing NFA

Example Find an NFA with a single final state that accepts the set $\{a\} \cup \{b^n | n \geq 1\}$.

Solution One solution is

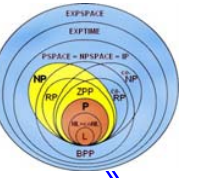




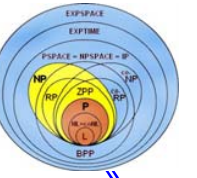
Why Nondeterminism

- Many deterministic algorithms require that one make a choice at some stage. A typical example is a game-playing program. Frequently, the best move is not known, but can be found using an exhaustive search with *backtracking*. A nondeterministic algorithm that can make the best choice would be able to solve the problem without backtracking.
- Nondeterminism is sometimes helpful in solving problems easily.
- There is a technical reason for introducing nondeterminism.


Break for 15 minutes



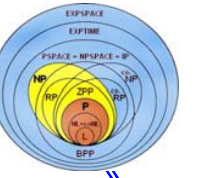
Equivalence of DFA and NFA



Equivalence of DFA and NFA

There are many languages for which an NFA is easier to construct than a DFA. That means NFA's usually easier to "program" in.

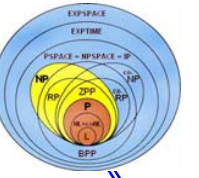
However, surprisingly, for any NFA N there is a DFA D such that $L(A) = L(N)$, and vise verse.



The proof that DFA's can do whatever NFA's can do involves an important "construction" called the subset construction, an important example how an automaton B can be generally constructed from another automaton A.

The subset construction starts from an NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$. Its goal is the description of a DFA $D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ such that

$$L(D) = L(N).$$

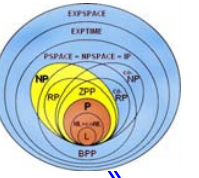


Here is the detail of the construction.

- $Q_D = \{S \mid S \subseteq Q_N\}$
- $F_D = \{S \mid S \subseteq Q_N, S \cap F_N \neq \emptyset\}$
- For every $S \subseteq Q_N$ and $a \in \Sigma$,

$$\delta_D(S, a) = \bigcup_{p \in S} \delta_N(p, a)$$

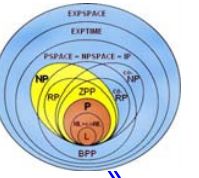
☞ Note that $|Q_D| = 2^{|Q_N|}$, although most states in Q_D are likely to be garbage.



Example Let's construct δ_D from NFA $A_2 = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\})$ where δ is the transition function

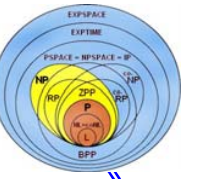
	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
$\star q_2$	$\emptyset$	$\emptyset$

Since N's set of states is $\{q_0, q_1, q_2\}$, the subset construction produces a DFA with $2^3 = 8$ states, corresponding to all the subsets of these three states. Next slide shows the transition table for these eight states.



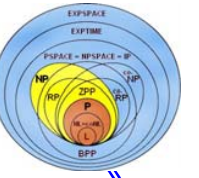
	0	1
$\emptyset$	$\emptyset$	$\emptyset$
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_1\}$	$\emptyset$	$\{q_2\}$
$\star\{q_2\}$	$\emptyset$	$\emptyset$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\star\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\star\{q_1, q_2\}$	$\emptyset$	$\{q_2\}$
$\star\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

To make the point clearer, we can invent new names for these states, e.g., A for $\emptyset$, B for $\{q_0\}$, and so on.



	0	1
A	A	A
$\rightarrow B$	E	B
C	A	D
$\star D$	A	A
E	E	F
$\star F$	E	B
$\star G$	A	D
$\star H$	E	F

Starting in the start state B, we can only reach states B, E, and F. The other five states are inaccessible from the start state and may as well not be there.

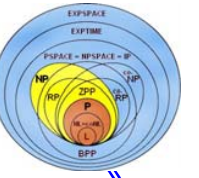


We can often avoid the exponential blow-up by constructing the transition table for D only foe accessible states S as follows (lazy evaluation):

Basis step: $S = \{q_0\}$ is accessible in D.

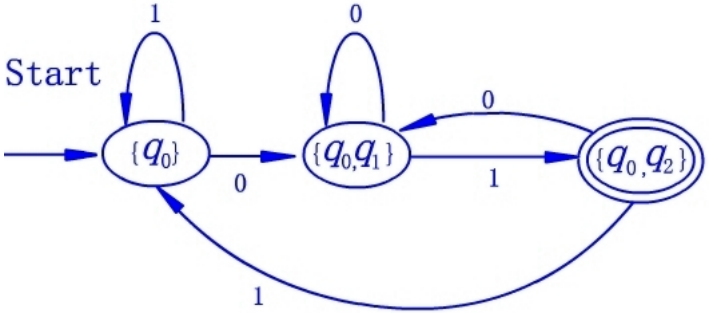
Inductive step: If state S is accessible, so are the states in

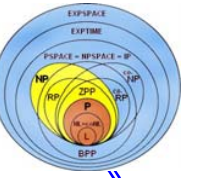
$$\bigcup_{a \in \Sigma} \delta_D(S, a).$$



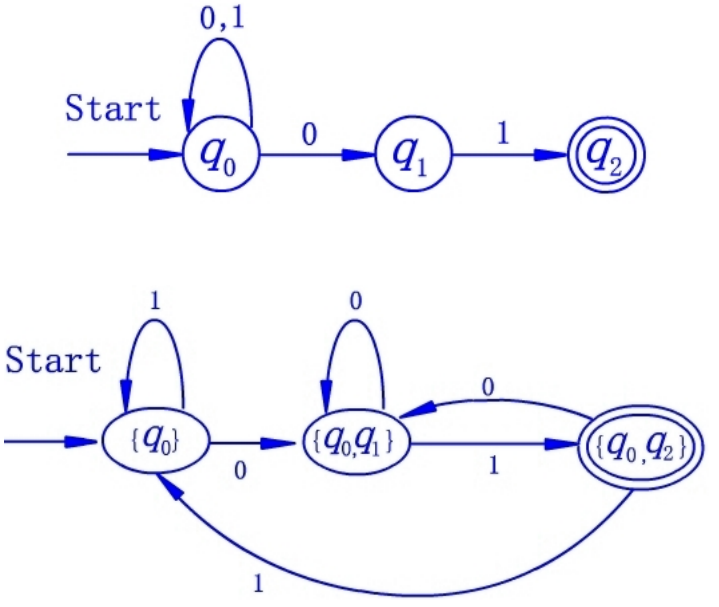
In this way, we obtain the "subset" DFA with accessible states only.

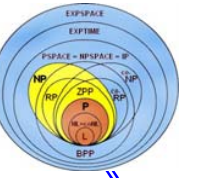
	0	1
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\star\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$





Comparing the DFA and NFA, we find that they have same number of states, but different number of transitions.



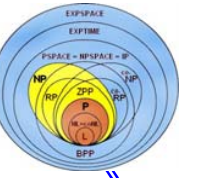


Although the intuition was suggested by the examples, we need to show formally that the subset construction works!

Theorem 2.1 *Let* $D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ *be the DFA constructed from NFA* $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ *by the subset construction, then* $L(A) = L(N)$.

Proof What we actually prove, by induction on $|w|$, is that

$$\hat{\delta}_D(\{q_0\}, w) = \hat{\delta}_N(q_0, w)$$

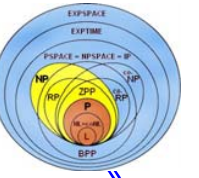


Basis step: Let $|w| = 0$, that is $w = \epsilon$. The claim follows from the definition of $\hat{\delta}$.

Inductive step: Let $|w| = n+1$, and assume the statement is true for length n. Break w as $w = xa$,

$$\begin{aligned}
\hat{\delta}_D(\{q_0\}, xa) &= \delta_D(\hat{\delta}_D(\{q_0\}, x), a) && \text{(definition of } \hat{\delta}_D\text{)} \\
&= \delta_D(\hat{\delta}_N(q_0, x), a) && \text{(induction hypothesis)} \\
&= \bigcup_{p \in \hat{\delta}_N(q_0, x)} \delta_N(p, a) && \text{(subset construction)} \\
&= \hat{\delta}_N(q_0, xa) && \text{(definition of } \hat{\delta}_N\text{)}
\end{aligned}$$

◀



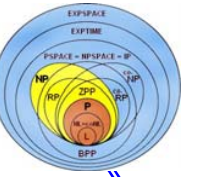
Theorem 2.2 *A language L is accepted by some DFA if and only if L is accepted by some NFA.*

Proof (if) This part is Theorem 2.1.

(only-if) Note that any DFA can be converted to an equivalent NFA by modifying the δ_D to δ_N by the rule

- If $\delta_D(q, a) = p$, then $\delta_N(q, a) = \{p\}$.

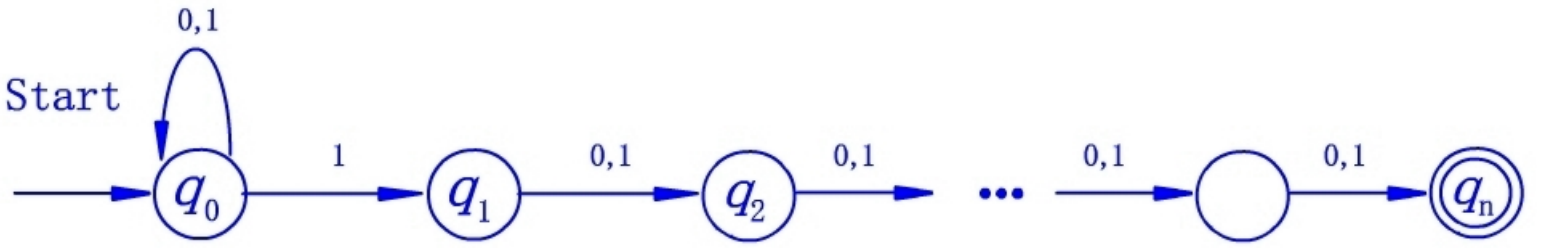
By induction on $|w|$, it will be shown in the tutorial that if $\hat{\delta}_D(q_0, w) = p$, then $\hat{\delta}_N(q_0, w) = \{p\}$. The claim of the theorem follows. ◄

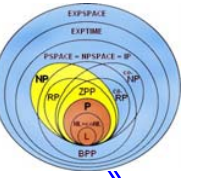


Bad Case for the Subset Construction

For DFA constructed by NFA, exponential growth in the number of states is possible.

Example There is an NFA N with $n + 1$ states that has no equivalent DFA with fewer than 2^n states.

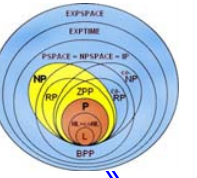




Clearly, $L(N) = \{x1c_2c_3 \cdots c_n \mid x \in \{0,1\}^*, c_i \in \{0,1\}\}$. Intuitively, a DFA D that accepts $L(N)$ must remember the last n symbols it has read.

Since there are 2^n bit sequences $a_1a_2 \cdots a_n$, if D with fewer than 2^n states exists, then there would be some state q such that D can be in state q after reading two different sequences of n bits, say $a_1a_2 \cdots a_n$ and $b_1b_2 \cdots b_n$.

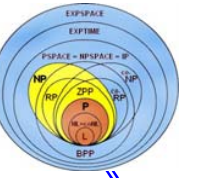
Since the sequences are different, there is i such that $a_i \neq b_i$. Suppose (by symmetry) that $a_i = 1$ and $b_i = 0$.



Two cases will be consider.

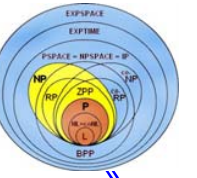
1. If $i = 1$, i.e., the sequences are $1a_2 \cdots a_n$ and $0b_2 \cdots b_n$, then q must be both an accepting state and a nonaccepting state.
2. If $i > 1$, i.e., the sequences are $a_1 \cdots a_{i-1}1a_{i+1} \cdots a_n$ and $b_1 \cdots b_{i-1}0b_{i+1} \cdots b_n$, then consider the state p that D enters after reading $i - 1$ 0's. Then p must be both accepting and nonaccepting.

The claim follows by contradiction.


EXPSPACE
EXPTIME
PSPACE = NPSPACE = IP
NP
RP
ZPP
P
NP
RP
L
BPP

Thank you!


華東師範大學
EAST CHINA NORMAL UNIVERSITY


EXPSPACE
EXPTIME
PSPACE = NPSPACE = IP
NP
RP
ZPP
P
L
BPP


華東師範大學
EAST CHINA NORMAL UNIVERSITY