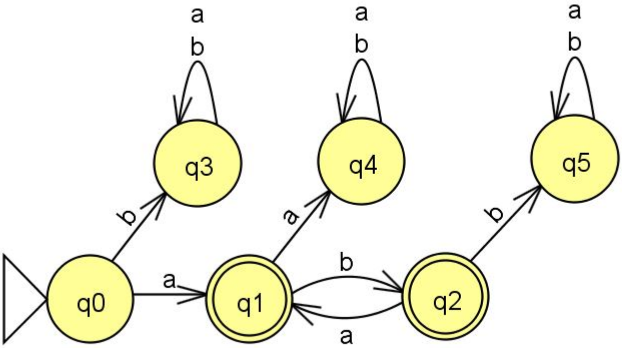
CMSC 141 Automata and Language Theory

Regular Languages

Mark Froilan B. Tandoc

August 20, 2014

Another look in our FA



Deterministic Finite Automata (DFA)

A DFA is (formally) defined using a "five-tuple" $M = (Q, \Sigma, \delta, S, F)$ where

- $Q \Rightarrow$ finite set of states, e.g. $\{q_0, q_1, q_2\}$
- $\Sigma \Rightarrow$ input alphabet, e.g. $\{a, b\}$
- $\delta \Rightarrow$ transition function, $Q \times \Sigma \rightarrow Q$
  $\delta(q_0, a) = q_1$
  $\delta(q_0, b) = q_2$
- $S \Rightarrow$ start state, where $S \in Q$
- $F \Rightarrow$ set of final(accepting) states, where $F \subseteq Q$

Simpler Notations for DFA's

Specifying DFA using five-tuple is both tedious and hard to read

five-tuple

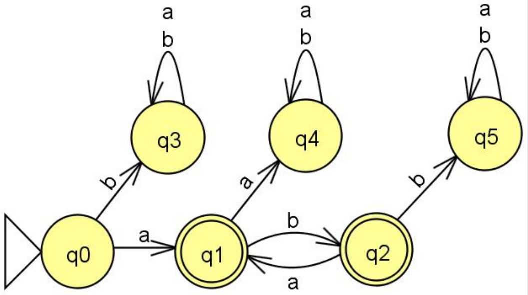
$M = (Q, \Sigma, \delta, q_0, F)$ where:

- $Q = \{q_0, q_1, q_2, ..., q_5\}$
- $\Sigma = \{a, b\}$
- $\delta(q_0, a) = q_1$
  $\delta(q_0, b) = q_3$
  $\delta(q_1, a) = q_4$
  $\delta(q_1, b) = q_2$
  ...
- $F = \{q_1, q_2\}$

Simpler Notations for DFA's

Transition diagram or transition table can be used as alternatives

transition diagram



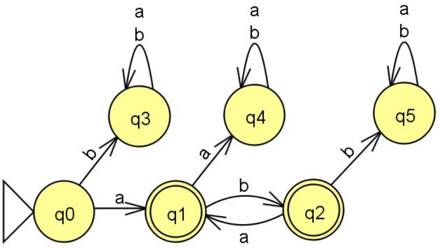
transition table

δ	a	b
$\rightarrow q_0$	q_1	q_3
$*q_1$	q_4	q_2
$*q_2$	q_1	q_5
q_3	q_3	q_3
q_4	q_4	q_4
q_5	q_4	q_4

Instantaneous Descriptions of DFAs

- We represent the status of an execution of a DFA with the pair *(currentState, remainingInput)*
- Such pair is known as an ID and provide a snapshot of the process
- The acceptance of a string is demonstrated by a sequence of such IDs

Instantaneous Descriptions of DFAs



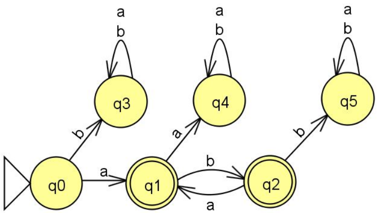
$\vdash$ "leads to"

$$
\begin{aligned}
(q_0, aba) &\vdash (q_1, ba) \\
&\vdash (q_2, a) \\
&\vdash (q_1, \varepsilon)
\end{aligned}
$$

$\vdash^*$ "eventually leads to"

$$(q_0, aba) \vdash^* (q_1, \varepsilon)$$

Exercises



Do we really need states q_3, q_4 and q_5?

Create DFAs for:

- $\{w|w$ has alternating symbols$\}$ over $\Sigma = \{a, b\}$
- $\{w|w$ has a substring 01$\}$ over $\Sigma = \{0, 1\}$
- $\{w|w$ has odd number of 1's$\}$ over $\Sigma = \{0, 1\}$
- $\{w|w$ has even number of 1's and 0's$\}$ over $\Sigma = \{0, 1\}$

Designing a DFA

- Whether automation or artwork, design is a creative process
- Put yourself in the place of the machine you are designing
- After every scanned symbol, you must decide if the string seen so far is in the language
- To make these decisions, you must remember some things about the string as you read them

Example

Design a DFA for $\{w|w$ has odd number of 1's$\}$ over $\Sigma = \{0, 1\}$

Designing a DFA

Example

Design a DFA for $\{w|w \text{ has odd number of 1's}\}$ over $\Sigma = \{0, 1\}$

- What do we need to remember?

Designing a DFA

Example

Design a DFA for $\{w|w \text{ has odd number of 1's}\}$ over $\Sigma = \{0, 1\}$

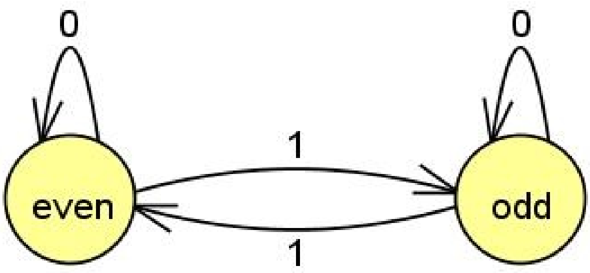
- We simply remember if the number of 1's seen so far is odd or even giving us two states

Designing a DFA

Example

Design a DFA for $\{w|w \text{ has odd number of 1's}\}$ over $\Sigma = \{0, 1\}$

- Now we assign transitions by changing states as we read the symbols

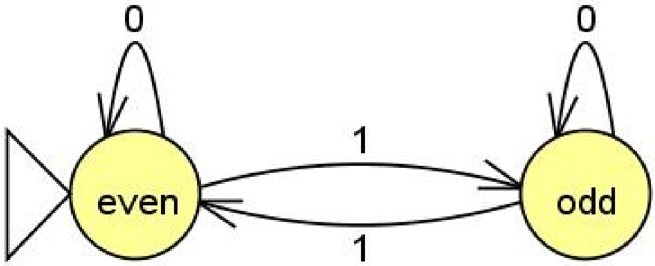


Designing a DFA

Example

Design a DFA for $\{w|w \text{ has odd number of 1's}\}$ over $\Sigma = \{0, 1\}$

- We set the start state where no symbol is seen so far

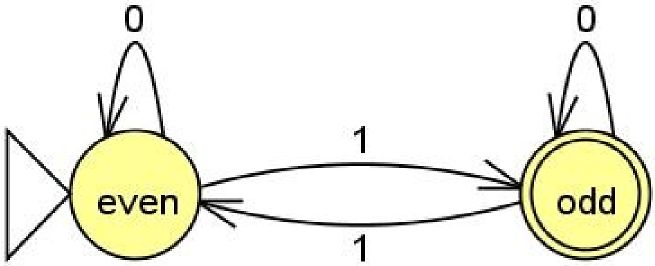


Designing a DFA

Example

Design a DFA for $\{w | w \text{ has odd number of 1's}\}$ over $\Sigma = \{0, 1\}$

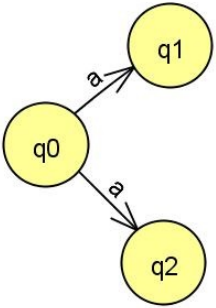
- Last, we set the states where we wanted to accept the strings



Non-deterministic Finite Automata (NFA)

- In DFAs, for each input, there is one and only one state to transition and we can only be in a single state in a given time
- NFAs allows us to be in multiple states simultaneously

- We can also look at it as if the machine creates a copy of itself on a different state
- In an NFA, a state may have zero, one, or many exiting arrows for each alphabet symbol

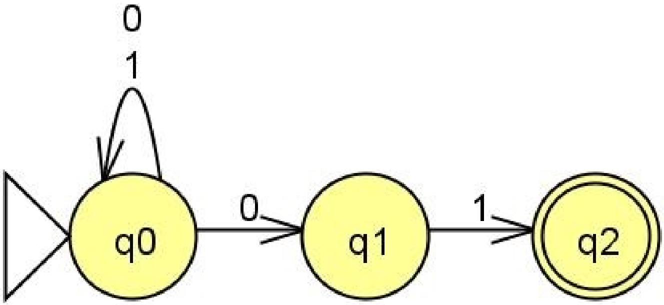


Acceptance in NFAs

- There would be a need to modify the definition of acceptance in NFAs
- A string x is accepted by an NFA if there is a path (one is enough) that eventually ends in a final state

Why the need for non-determinism?

- Use of non-determinism often simplifies the design of FA
- Consider $L = \{w | w \text{ ends in } 01\}$

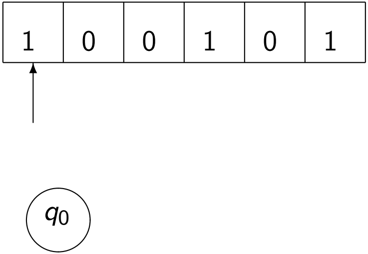


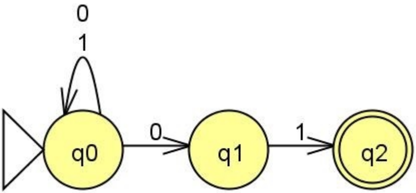
- Challenge: create a DFA of the language

Sample Run

$L = \{w|w \text{ ends in } 01\}$

Test if "100101" belong to the language

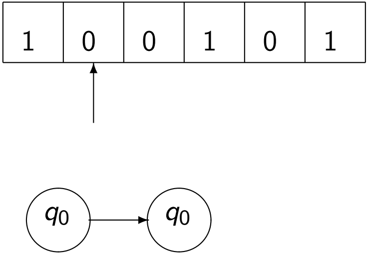


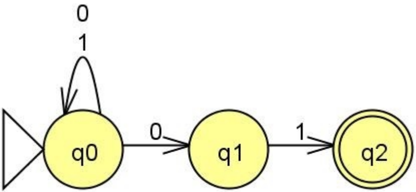


Sample Run

$L = \{w | w \text{ ends in } 01\}$

Test if "100101" belong to the language

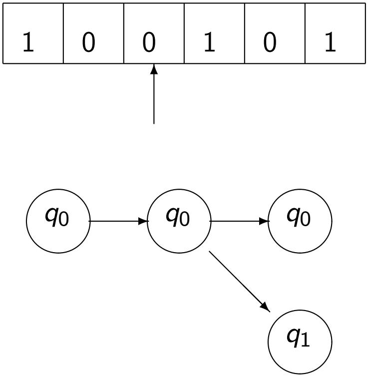


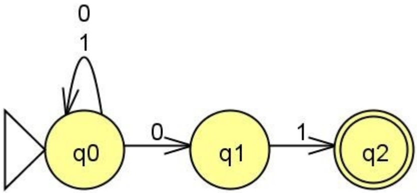


Sample Run

$L = \{w | w \text{ ends in } 01\}$

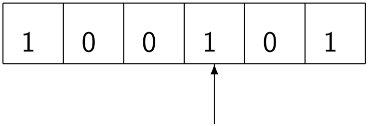
Test if "100101" belong to the language

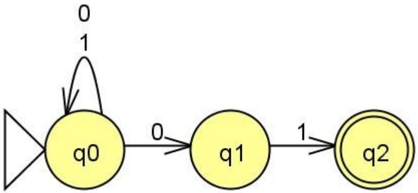


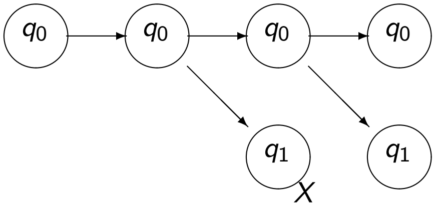


Sample Run

$L = \{w | w \text{ ends in } 01\}$
Test if "100101" belong to the language

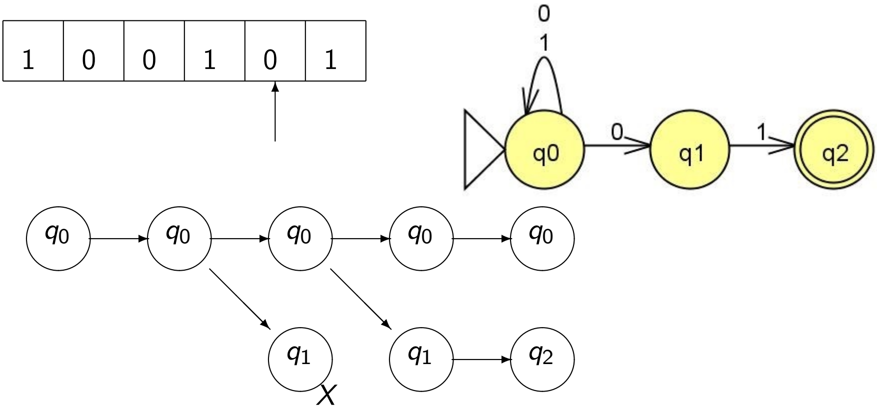






Sample Run

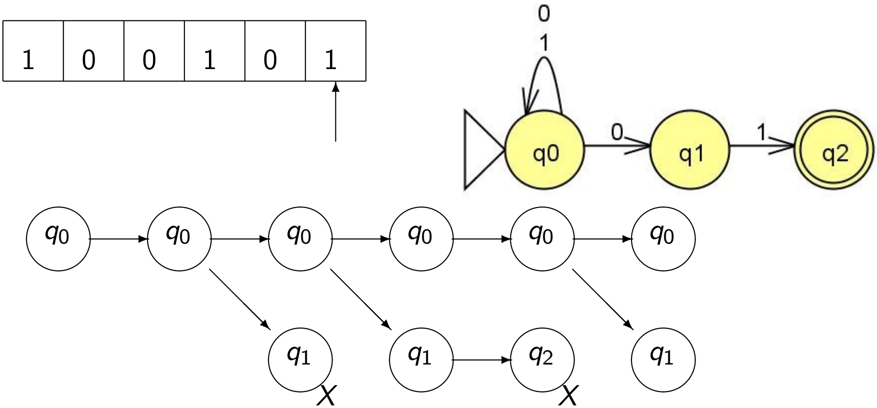
$L = \{w|w \text{ ends in } 01\}$
Test if "100101" belong to the language



Sample Run

$L = \{w | w \text{ ends in } 01\}$

Test if "100101" belong to the language

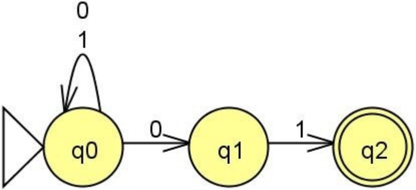


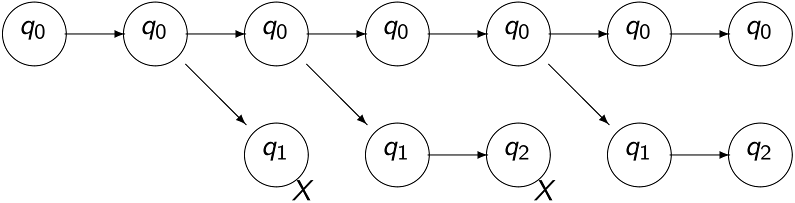
Sample Run

$L = \{w | w \text{ ends in } 01\}$

Test if "100101" belong to the language

1	0	0	1	0	1



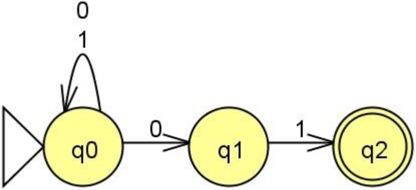


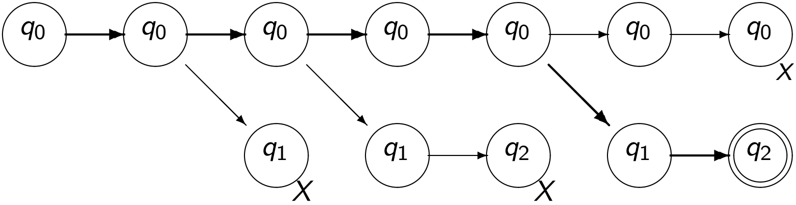
Sample Run

$L = \{w | w \text{ ends in } 01\}$

Test if "100101" belong to the language

1	0	0	1	0	1





Formal Definition of an NFA

NFAs are defined almost the same as DFAs using a "five-tuple" $M = (Q, \Sigma, \delta, S, F)$ where

- $Q \Rightarrow$ finite set of state, e.g. $\{q_0, q_1, q_2\}$
- $\Sigma \Rightarrow$ input alphabet, e.g. $\{0, 1\}$
- $\delta \Rightarrow$ transition function, $Q \times \Sigma \rightarrow P(Q)$
  The transition function takes a state and an input symbol and produces the set of possible next states e.g.

δ	0	1
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
q_2	$\emptyset$	$\emptyset$

- $S \Rightarrow$ start state, where $S \in Q$ e.g. q_0
- $F \Rightarrow$ set of final states, where $F \subseteq Q$ e.g. $\{q_2\}$

NFA $\Leftrightarrow$ DFA

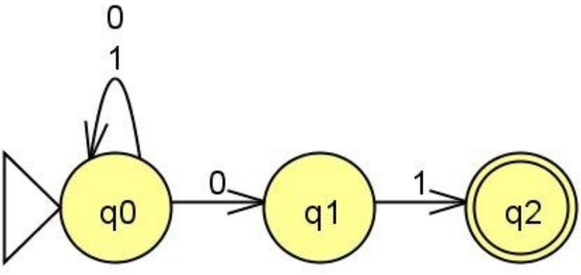
- All DFAs are NFAs.
- Can every NFA be converted to an equivalent DFA?

NFA $\Leftrightarrow$ DFA

- All DFAs are NFAs.
- Can every NFA be converted to an equivalent DFA?
- DFA and NFA recognize the same class of languages (regular languages).

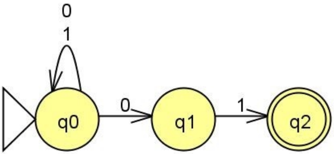
Converting an NFA into a DFA

Consider the following NFA

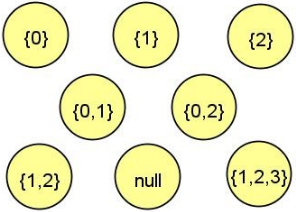


- Let $M = \{Q, \Sigma, \delta, q_0, F\}$ be the given NFA
- We construct an equivalent DFA $M' = \{Q', \Sigma, \delta', q_0', F'\}$

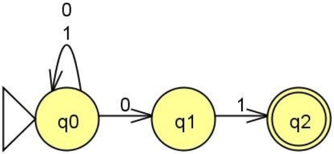
Converting an NFA into a DFA



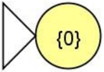
Every state in M' is a set of states from the NFA.



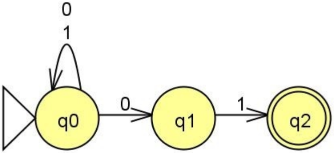
Converting an NFA into a DFA



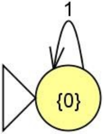
Start from the initial state where $q_0' = \{0\}$



Converting an NFA into a DFA

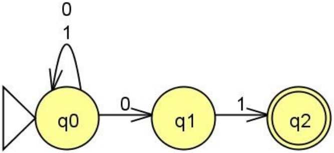


We build up the DFA state by state using the rules $\delta'(q', a) = \bigcup_{q \in q'} \delta(q, a)$, for all $q' \in Q', a \in \Sigma$

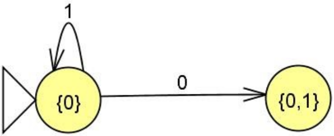


$$\delta'(\{0\}, 1) = \{0\}$$

Converting an NFA into a DFA

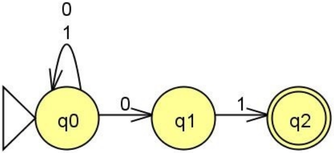


We build up the DFA state by state using the rules $\delta'(q', a) = \bigcup_{q \in q'} \delta(q, a)$, for all $q' \in Q', a \in \Sigma$

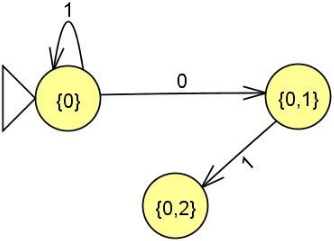


$$\delta'(\{0\}, 0) = \{0, 1\}$$

Converting an NFA into a DFA



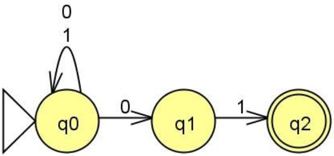
We build up the DFA state by state using the rules $\delta'(q', a) = \bigcup_{q \in q'} \delta(q, a)$, for all $q' \in Q', a \in \Sigma$



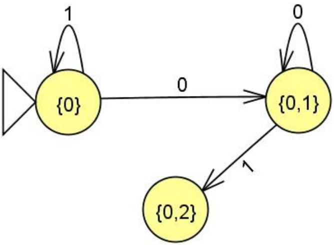
$$\delta'(\{0,1\},1) = \delta(0,1) \cup \delta(1,1)$$

$$\delta'(\{0,1\},1) = \{0\} \cup \{2\}$$

Converting an NFA into a DFA



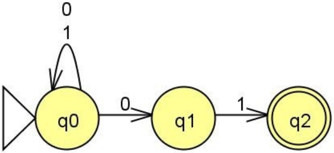
We build up the DFA state by state using the rules $\delta'(q', a) = \bigcup_{q \in q'} \delta(q, a)$, for all $q' \in Q', a \in \Sigma$



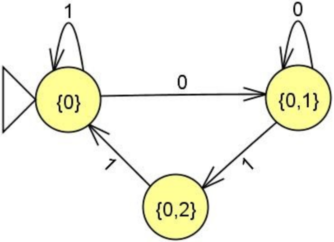
$$\delta'(\{0,1\}, 0) = \\ \delta(0,0) \cup \delta(1,0)$$

$$\delta'(\{0,1\}, 0) = \\ \{0,1\} \cup \emptyset$$

Converting an NFA into a DFA



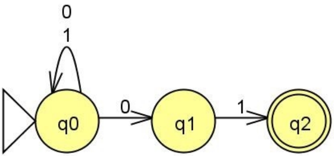
We build up the DFA state by state using the rules $\delta'(q', a) = \bigcup_{q \in q'} \delta(q, a)$, for all $q' \in Q', a \in \Sigma$



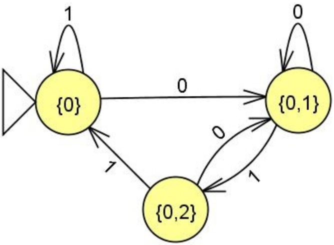
$$\delta'(\{0,2\},1) = \\ \delta(0,1) \cup \delta(2,1)$$

$$\delta'(\{0,2\},1) = \\ \{0\} \cup \emptyset$$

Converting an NFA into a DFA



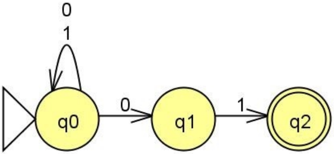
We build up the DFA state by state using the rules $\delta'(q', a) = \bigcup_{q \in q'} \delta(q, a)$, for all $q' \in Q', a \in \Sigma$



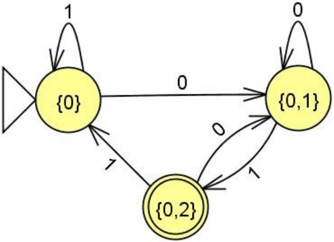
$$\delta'(\{0, 2\}, 0) = \delta(0, 0) \cup \delta(2, 0)$$

$$\delta'(\{0, 2\}, 0) = \{0, 1\} \cup \emptyset$$

Converting an NFA into a DFA



If a state in M' is a final state if it contains at least one final state from the NFA.



References

- Previous slides on CMSC 141
- M. Sipser. Introduction to the Theory of Computation. Thomson, 2007.
- J.E. Hopcroft, R. Motwani and J.D. Ullman. Introduction to Automata Theory, Languages and Computation. 2nd ed, Addison-Wesley, 2001.
- E.A. Albacea. Automata, Formal Languages and Computations, UPLB Foundation, Inc. 2005
- JFLAP, www.jflap.org