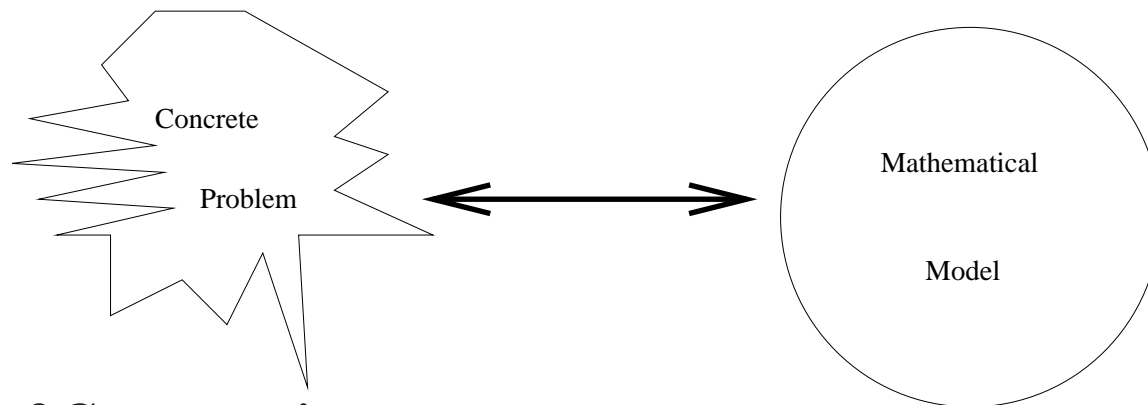


In-depth introduction to main models, concepts of theory of computation:

- **Automata Theory:** warm-up and review
- **Computability:** what can be computed in principle
- **Logic:** how can we express our requirements
- **Complexity:** what can be computed in practice



Formal Models of Computation:

- FA \equiv Regular Language
- PDA \equiv CFL
- TM = all powerful computer
- logical formula

deep understanding of formal models
of computation

proofs are important

one required text: available at Jeffery Amherst College

[P]: Christos Papadimitriou, *Computational Complexity*

Prerequisites: Mathematical maturity: reason abstractly, understand and write proofs.

Work:

- weekly problem sets (34% of grade)
- in-class Midterm (March 9, 2009) (33% of grade)
- final (TBA) (33% of grade)

Cooperation: Students should talk to each other and help each other; but **write up solutions on your own, in your own words**. Sharing or copying a solution could result in failure. If a significant part of one of your solutions is due to someone else, or something you've read then **you must acknowledge your source!**

Mathematical Sophistication

- Velleman, *How to Prove It*, 1994.

Review of Regular and Context-Free Languages

- Hopcroft, Motwani, and Jeffrey D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, 2001.
- Lewis and Papadimitriou, *Elements of the Theory of Computation*, 1998.
- Sipser, *Introduction to the Theory of Computation*, 1997.

Complexity

- Garey and Johnson, *Computers and Intractability*, 1979.
- Immerman, *Descriptive Complexity*, 1999.
- Kozen, *Theory of Computation*, 2006.

Lots of Info on Course Website: www.cs.umass.edu/~immerman/cs601

Definition: An **alphabet** is a non-empty finite set, e.g., $\Sigma = \{0, 1\}$, $\Gamma = \{a, b, c\}$, etc.

Definition: The set of **regular expressions** $R(\Sigma)$ over alphabet Σ is the smallest set of strings such that:

1. if $a \in \Sigma$ then $a \in R(\Sigma)$
2. $\emptyset \in \mathbf{R}(\Sigma)$
3. if $e, f \in R(\Sigma)$ then so are the following:
 - (a) $(e \cup f)$
 - (b) $(e \circ f)$
 - (c) (e^*)

Note: $\epsilon \in \mathbf{R}(\Sigma)$ but we don't need to mention this because $\mathcal{L}(\emptyset^*) = \mathcal{L}(\epsilon) = \{\epsilon\}$.

(The advantage of not including ϵ as one of the base cases is that we then have one fewer case when we prove things by induction about regular expressions.)

Examples:

- $e_1 = 0^* \in R(\{0, 1\})$
- $e_2 = ((a \cup b) \circ (a \cup b))^* \in R(\{a, b\})$
- $e_3 = a^*(ba^*ba^*)^* \in R(\{a, b, c\})$

Meanings:

- $\mathcal{L}(0^*) = \{\epsilon, 0, 00, 0^3, 0^4, \dots\} = \{0^i \mid i \in \mathbf{N}\}$
- $\mathcal{L}((a \cup b)^{2*}) = \{w \in \{a, b\}^* \mid |w| \equiv 0 \pmod{2}\}$
- $\mathcal{L}(a^*(ba^*ba^*)^*) = \{w \in \{a, b\}^* \mid \#_b(w) \equiv 0 \pmod{2}\}$

Recall the meaning of Kleene star, for any set, A ,

$$\begin{aligned} A^* &\equiv \bigcup_{i=0}^{\infty} A^i \\ &\equiv A^0 \cup A^1 \cup A^2 \cup \dots \\ &\equiv \{\epsilon\} \cup A \cup \{xy \mid x, y \in A\} \cup \dots \\ &\equiv \{x_1x_2 \dots x_n \mid n \in \mathbf{N}; x_1, \dots, x_n \in A\} \end{aligned}$$

Meaning of a Regular Expression:

1. if $a \in \Sigma$ then $a \in R(\Sigma)$; $\mathcal{L}(a) = \{a\}$
2. $\emptyset \in \mathbf{R}(\Sigma)$; $\mathcal{L}(\emptyset) = \emptyset$
3. if $e, f \in R(\Sigma)$ then so are $(e \cup f)$, $(e \circ f)$, (e^*) :

$$\mathcal{L}(e \cup f) = \mathcal{L}(e) \cup \mathcal{L}(f)$$

$$\mathcal{L}(e \circ f) = \mathcal{L}(e)\mathcal{L}(f) = \{uv \mid u \in \mathcal{L}(e), v \in \mathcal{L}(f)\}$$

$$\mathcal{L}(e^*) = (\mathcal{L}(e))^*$$

Definition 1.1 A set, A , is **regular** iff there exists a regular expression that denotes it. In symbols, the set of **regular sets** over Σ is,

$$\text{Regular}(\Sigma) = \{\mathcal{L}(e) \mid e \in R(\Sigma)\}$$

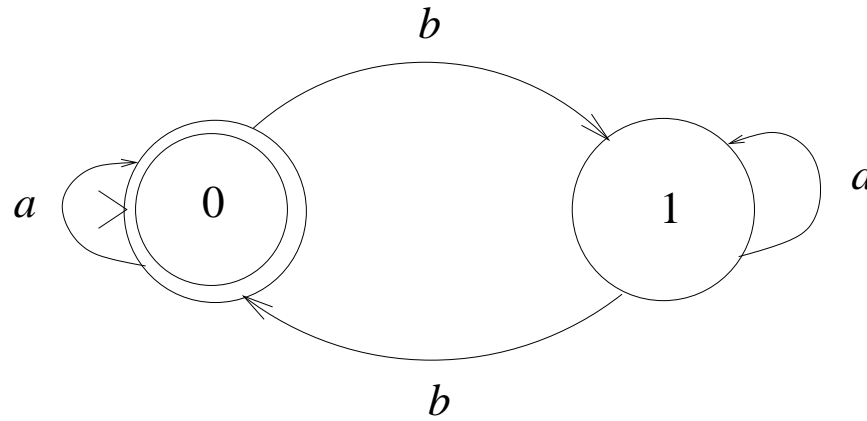
□

Definition: A **deterministic finite automaton (DFA)** is a tuple,

$$D = (Q, \Sigma, \delta, s, F)$$

- Q is a finite set of states,
- Σ is a finite alphabet,
- $\delta : Q \times \Sigma \rightarrow Q$ is the transition function,
- $s \in Q$ is the start state, and
- $F \subseteq Q$ is the set of final or accept states.

$$D_1 = (\{0, 1\}, \{a, b\}, \delta_1, 0, \{0\}) \quad \delta_1 = \{\langle\langle 0, a \rangle, 0 \rangle, \langle\langle 0, b \rangle, 1 \rangle, \langle\langle 1, a \rangle, 1 \rangle, \langle\langle 1, b \rangle, 0 \rangle\}$$



	<i>a</i>	<i>a</i>	<i>b</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>a</i>	<i>a</i>	<i>b</i>	<i>a</i>	
	0	0	0	1	0	0	1	1	1	0	0

$$\mathcal{L}_1 = \mathcal{L}(D_1) = \{w \in \{a, b\}^* \mid \#_b(w) \equiv 0 \pmod{2}\} = \mathcal{L}(a^*(ba^*ba^*)^*)$$

Definition: A nondeterministic finite automaton (NFA) is a tuple,

$$N = (Q, \Sigma, \Delta, s, F)$$

- Q is a finite set of states,
- Σ is a finite alphabet,
- $\Delta : (Q \times (\Sigma \cup \{\epsilon\})) \rightarrow \wp(Q)$ is the transition function,
- $s \in Q$ is the start state, and
- $F \subseteq Q$ is the set of final or accept states.

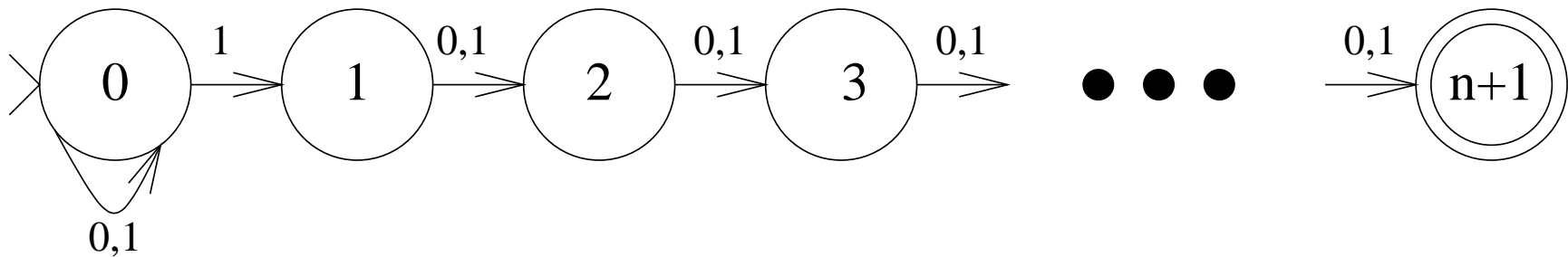
$$\mathcal{L}(N) = \{w \mid s \xrightarrow[w]{*} q \in F\}$$

$s \xrightarrow[w]{*} q$ means that there exists a path from s to q that reads w .

$$\wp(S) = \text{power set of } S = \{A \mid A \subseteq S\}$$

$$N_n = (\{q_0, \dots, q_{n+1}\}, \{0, 1\}, \Delta_n, q_0, \{q_{n+1}\})$$

$$\Delta_n = \{\langle\langle q_0, 0 \rangle, \{q_0\}\rangle, \langle\langle q_0, 1 \rangle, \{q_0, q_1\}\rangle, \dots, \langle\langle q_n, 1 \rangle, \{q_{n+1}\}\rangle\}$$

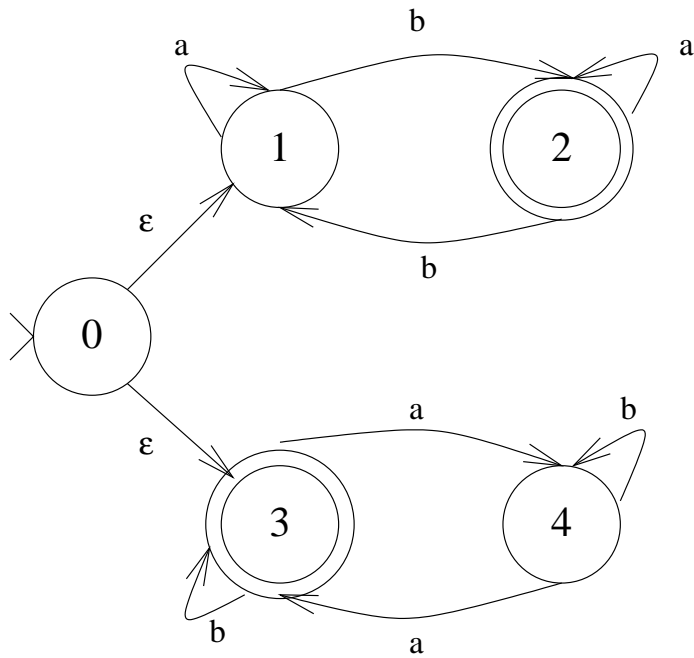


We will see later: The minimal DFA that accepts $\mathcal{L}(N_n)$ has 2^{n+1} states.

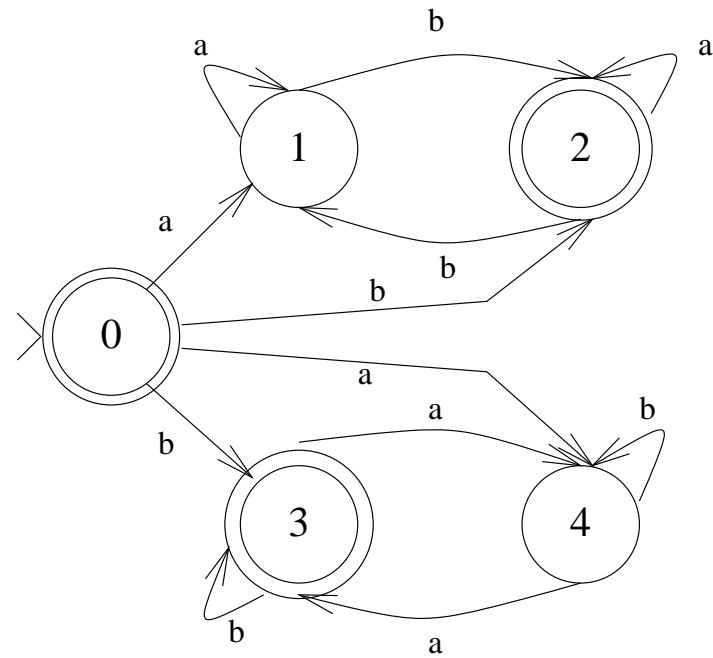
Proposition 1.2 Every NFA N can be translated into an NFA wo ϵ -transitions N' s.t. $\mathcal{L}(N) = \mathcal{L}(N')$

Proof: Given $N = (Q, \Sigma, \Delta, q_0, F)$, let $N' = (Q, \Sigma, \Delta', q_0, F')$ where

$$\Delta'(q, a) = \{r \mid (\exists s, t) q \xrightarrow{\epsilon^*} s \xrightarrow{a} t \xrightarrow{\epsilon^*} r\} \quad \mathbf{F}' = \{q \mid (\exists s \in F) q \xrightarrow{\epsilon^*} s\}$$



N



N'

□

Notation: For a DFA, $D = (Q, \Sigma, \delta, s, F)$, let $\delta^*(q, w)$ be the state that D will be in after reading string w , when started in q ,

$$\delta^*(q, \epsilon) \equiv q$$

$$\delta^*(q, wa) \equiv \delta(\delta^*(q, w), a)$$

$$\mathcal{L}(D) \equiv \{w \mid \delta^*(s, w) \in F\}$$

For an NFA without ϵ transitions, $N = (Q, \Sigma, \Delta, s, F)$, let $\Delta^*(q, w)$ be the set of states that N can be in after reading string w , when started in q ,

$$\Delta^*(q, \epsilon) := \{q\}$$

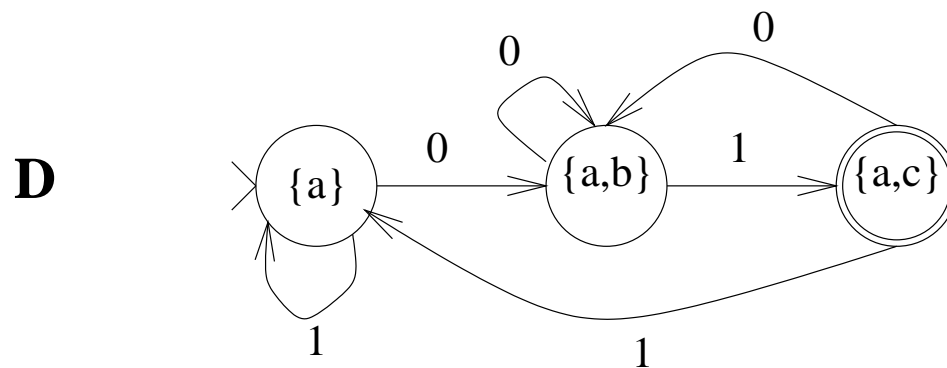
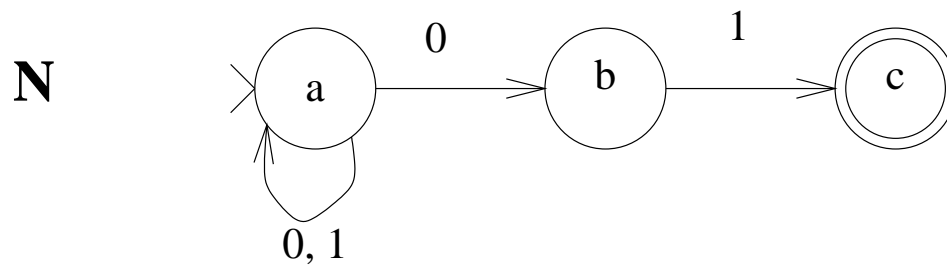
$$\Delta^*(q, wa) := \bigcup_{r \in \Delta^*(q, w)} \Delta(r, a)$$

$$\mathcal{L}(N) \equiv \{w \mid \Delta^*(s, w) \cap F \neq \emptyset\}$$

Proposition 1.3 For every NFA, N , with n states, there is a DFA, D , with at most 2^n states s.t. $\mathcal{L}(D) = \mathcal{L}(N)$.

Proof: Let $N = (Q, \Sigma, \Delta, q_0, F)$. By Proposition 1.2 may assume that N has no ϵ transitions. Let $D = (\wp(Q), \Sigma, \delta, \{q_0\}, F')$

$$\delta(S, a) = \bigcup_{r \in S} \Delta(r, a) \quad F' = \{S \subseteq Q \mid S \cap F \neq \emptyset\}$$



Claim: For all $w \in \Sigma^*$,

$$\delta^*(\{q_0\}, w) = \Delta^*(q_0, w)$$

By induction on $|w|$:

$$|w| = 0: \quad \delta^*(\{q_0\}, \epsilon) = \{q_0\} = \Delta^*(q_0, \epsilon)$$

$$|w| = k + 1: \quad w = ua.$$

Inductively, assume: $\delta^*(\{q_0\}, u) = \Delta^*(q_0, u)$

$$\begin{aligned} \delta^*(\{q_0\}, ua) &= \delta(\delta^*(\{q_0\}, u), a) \\ &= \bigcup_{r \in \delta^*(\{q_0\}, u)} \Delta(r, a) \\ &= \bigcup_{r \in \Delta^*(q_0, u)} \Delta(r, a) \\ &= \Delta^*(q_0, ua) \end{aligned}$$

Therefore, $\mathcal{L}(D) = \mathcal{L}(N)$. □

Theorem 1.4 (Kleene's Th) Let $A \subseteq \Sigma^*$ be any language. Then the following are equivalent: 1. $A = \mathcal{L}(D)$, for some DFA D .

2. $A = \mathcal{L}(N)$, for some NFA N w/o ϵ transitions

3. $A = \mathcal{L}(N)$, for some NFA N .

4. $A = \mathcal{L}(e)$, for some regular expression e .

5. A is regular.

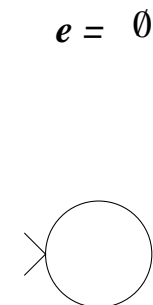
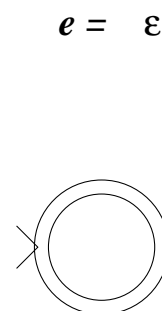
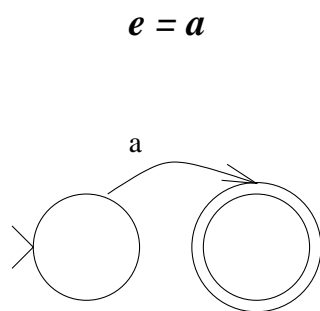
Proof: Obvious that $1 \rightarrow 2 \rightarrow 3$.

$3 \rightarrow 2$ by Prop. 1.2.

$2 \rightarrow 1$ by Prop. 1.3 (subset construction).

$4 \leftrightarrow 5$ by def of regular

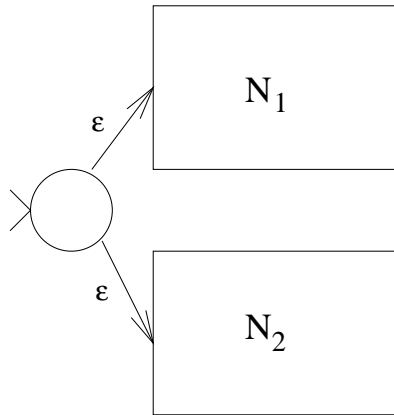
$4 \rightarrow 3$: We will show by induction on the number of symbols in the regular expression e , that there is an NFA N with $\mathcal{L}(e) = \mathcal{L}(N)$. **Base Case:**



Inductive Cases:

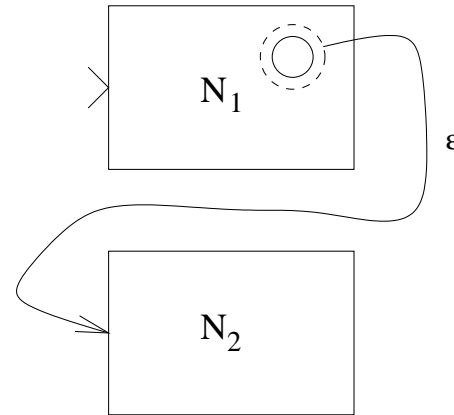
Union

$$L(N) = L(N_1) + L(N_2)$$



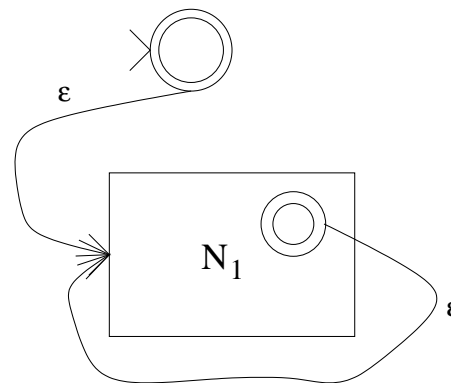
Concatenation

$$L(N) = L(N_1) L(N_2)$$



Kleene Star

$$L(N) = (L(N_1))^*$$



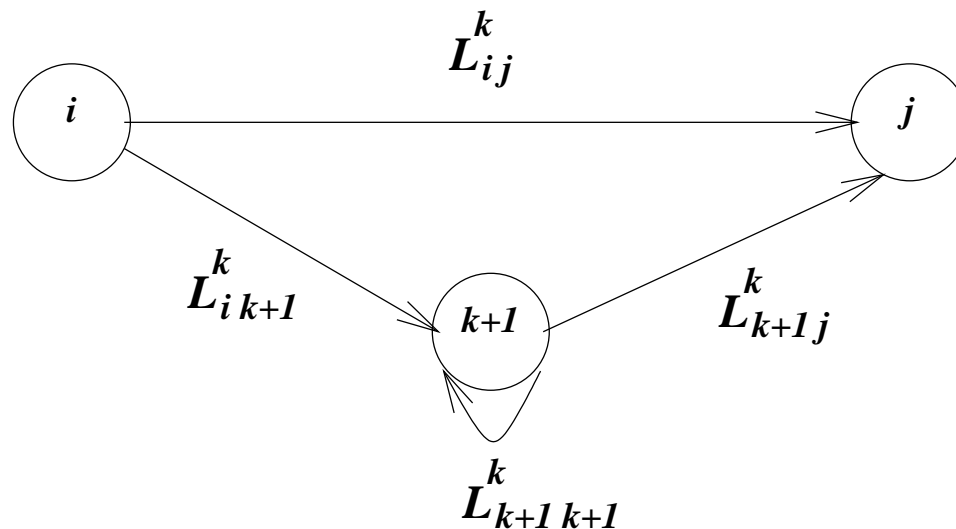
3 \rightarrow 4 (NFA implies reg exp): Let $N = (\{1, \dots, n\}, \Sigma, \Delta, 1, F)$, $F = \{f_1, \dots, f_r\}$

Dynamic Programming Algorithm:

$$L_{ij}^k \equiv \{w \mid i \xrightarrow[w]{*} j; \text{ no intermediate state } \# > k\}$$

$$L_{ij}^0 = \{a \mid j \in \Delta(i, a)\} \cup \{\epsilon \mid i = j\}$$

$$L_{ij}^{k+1} = L_{ij}^k \cup L_{ik+1}^k (L_{k+1 k+1}^k)^* L_{k+1 j}^k$$



3 \rightarrow 4 (NFA implies reg exp): Let $N = (\{1, \dots, n\}, \Sigma, \Delta, 1, F)$, $F = \{f_1, \dots, f_r\}$

Dynamic Programming Algorithm:

$$L_{ij}^k \equiv \{w \mid i \xrightarrow[w]{*} j; \text{ no intermediate state } \# > k\}$$

$$L_{ij}^0 = \{a \mid j \in \Delta(i, a)\} \cup \{\epsilon \mid i = j\}$$

$$L_{ij}^{k+1} = L_{ij}^k \cup L_{ik+1}^k (L_{k+1, k+1}^k)^* L_{k+1, j}^k$$

$$e = L_{1f_1}^n \cup \dots \cup L_{1f_r}^n$$

$$\mathcal{L}(e) = \mathcal{L}(N)$$

□

