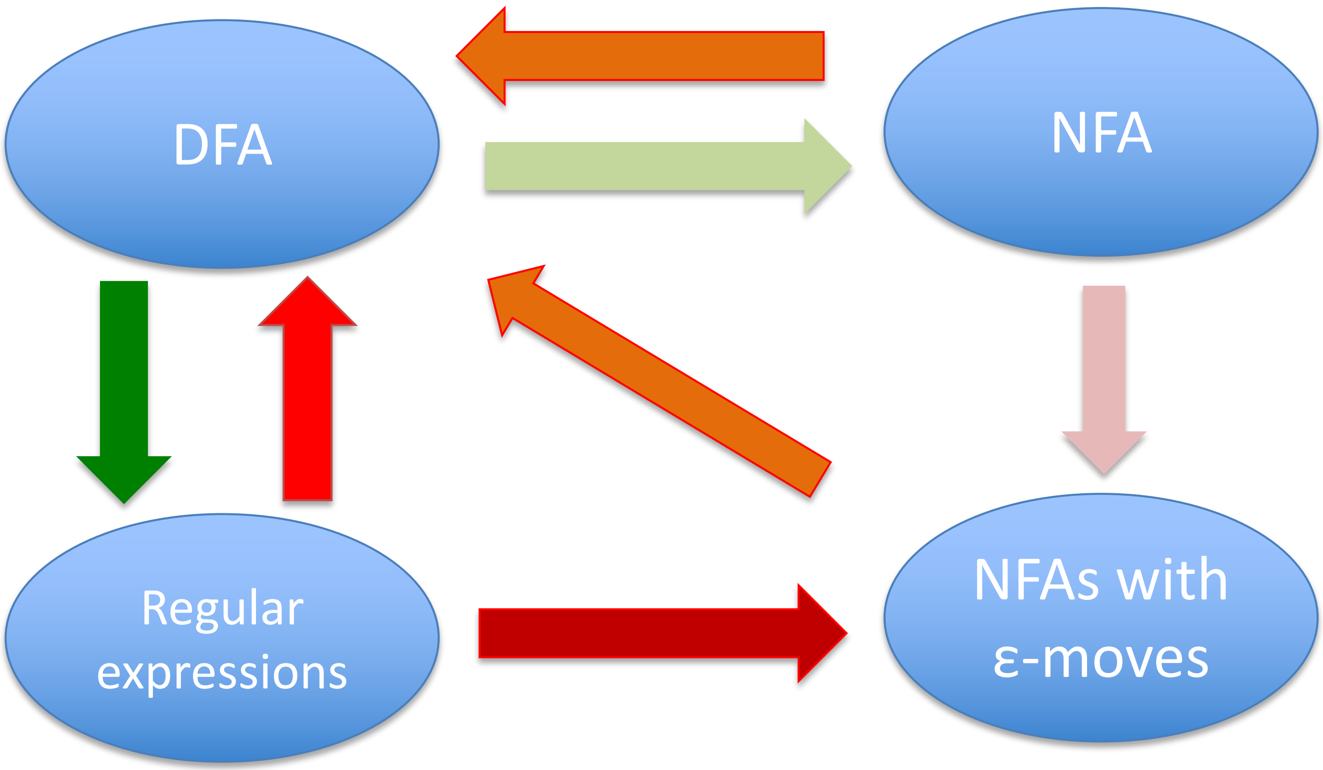
4.

EQUIVALENCE RESULTS FOR REGULAR LANGUAGES

The big picture



REVIEW – FROM LAST LECTURES

Regular expressions and their language

- $\emptyset$ $\quad L(\emptyset) = \{\ \}$
- ε $\quad L(\varepsilon) = \{\varepsilon\}$
- A $\quad L(a) = \{a\}$
- $E+F$ $\quad L(E+F) = L(E) \cup L(F)$
- $E\ F$ $\quad L(E\ F) = L(E)\ L(F)$
- E^* $\quad L(E^*) = L(E)^*$

DFAs and NFAs and their languages

$\hat{\delta}$ is written $\underline{\delta}$ on these slides

$(Q, \Sigma, \delta, q_0, F)$

- where $\delta: Q \times \Sigma \rightarrow Q$

Accepts the language

- $\{w \mid \underline{\delta}(q_0,w)$ is in $F\}$

$\underline{\delta}(q,\varepsilon) = q$
$\underline{\delta}(q,wa) = \delta(\underline{\delta}(q,w),a)$

$(Q, \Sigma, \delta, q_0, F)$

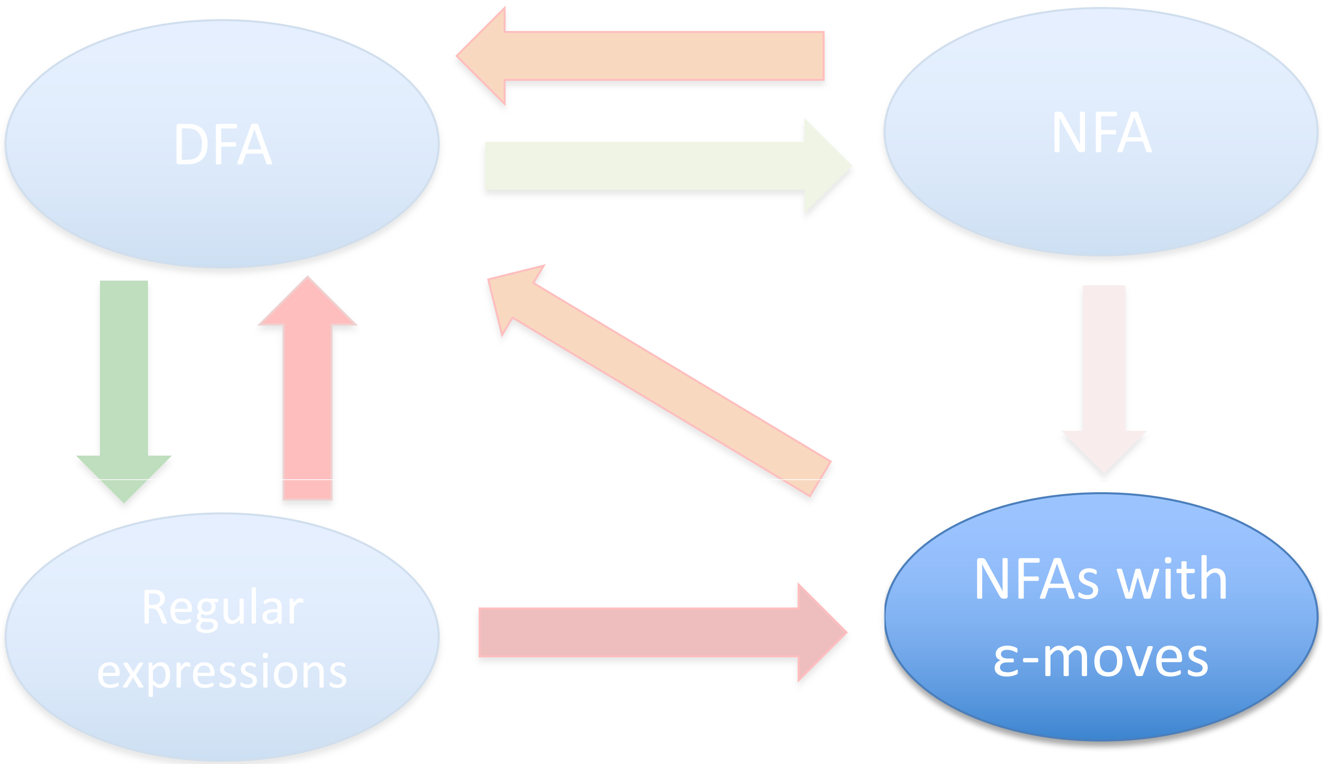
- where $\delta: Q \times \Sigma \rightarrow \boldsymbol{P}(Q)$

Accepts the language

- $\{w \mid \underline{\delta}(q_0,w)$ contains a state from $F\}$

$\underline{\delta}(q,\varepsilon) = \{q\}$
$\underline{\delta}(q,wa) = \delta(q_1,a) \cup \ldots \cup \delta(q_k,a)$
where $\underline{\delta}(q,w) = \{q_1,\ldots,q_k\}$

The big picture



Formal definition

A non-deterministic finite automaton with ε-moves consists of

- Q: a finite set of states
- Σ: a finite set of input symbols, an alphabet
- $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow \mathcal{P}(Q)$: a transition function

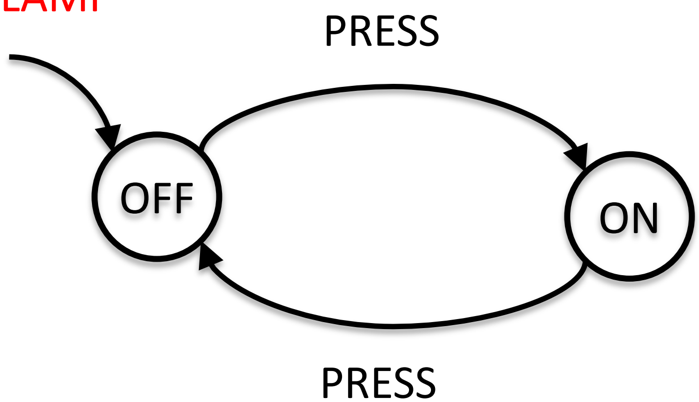
  for each state q in Q and perhaps a symbol a in Σ it determines a set of new states $\delta(q,a)$
- q_0: the initial state; an element of Q
- F: the final states; a subset of Q

The automaton does not need to read a symbol

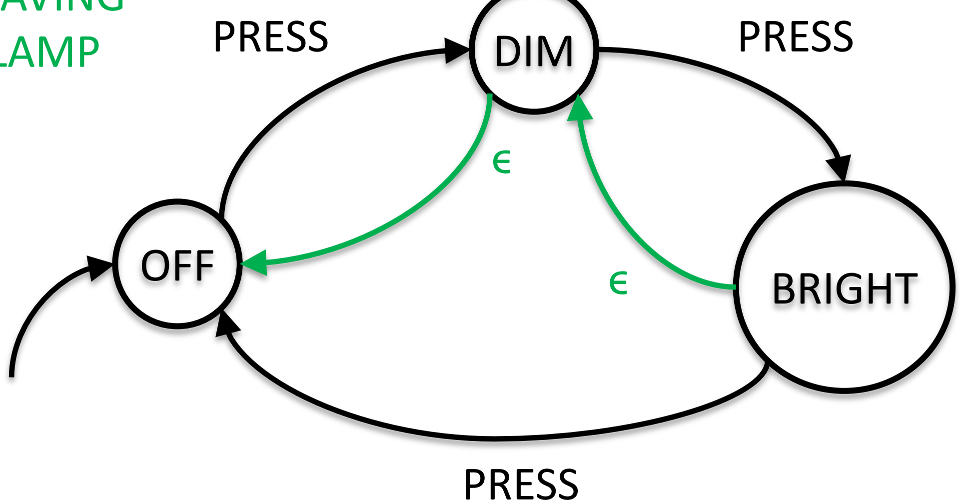
Often written as $A = (Q, \Sigma, \delta, q_0, F)$

Example: A lamp

BASIC LAMP



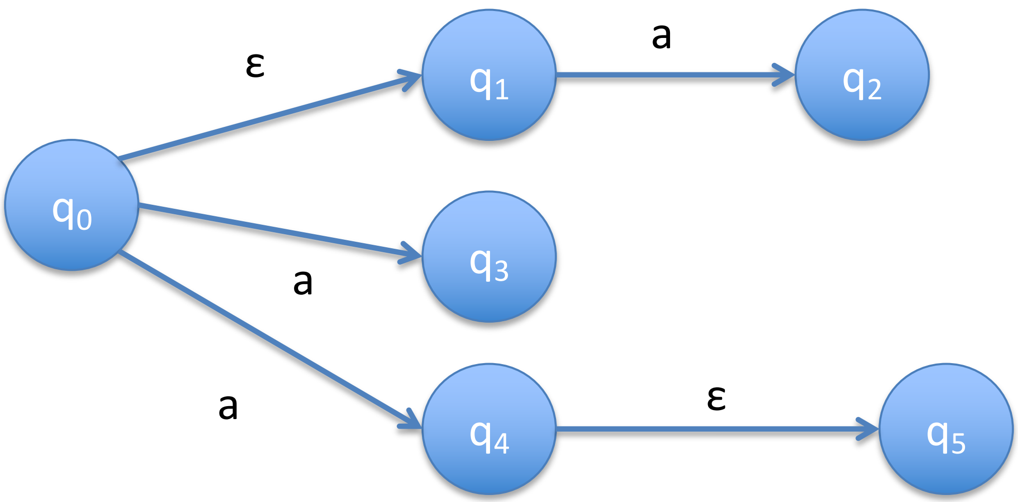
ENERGY SAVING
SMART LAMP



ϵ are ”internal actions”

The language accepted by an ε-NFA

- How can we define the extended transition function $\underline{\delta}$? I.e. $\underline{\delta}(q_0,a) = ?$

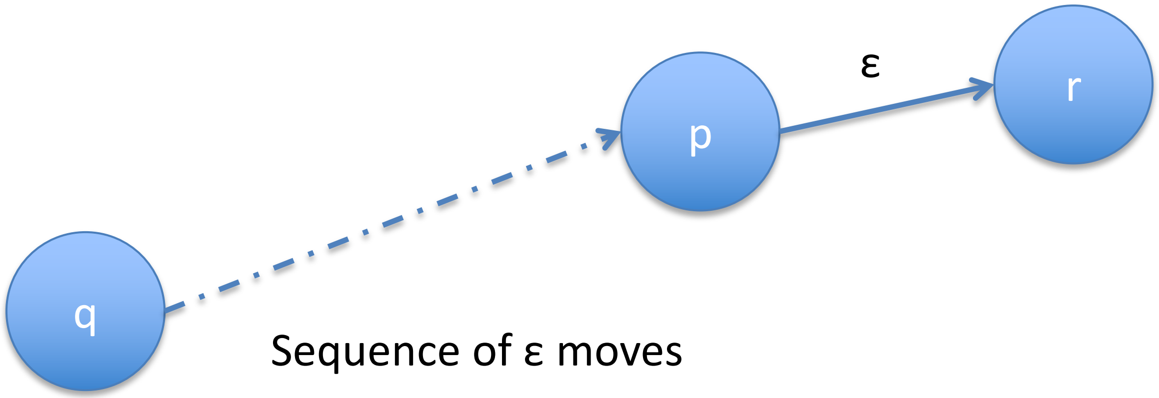


IDEA:

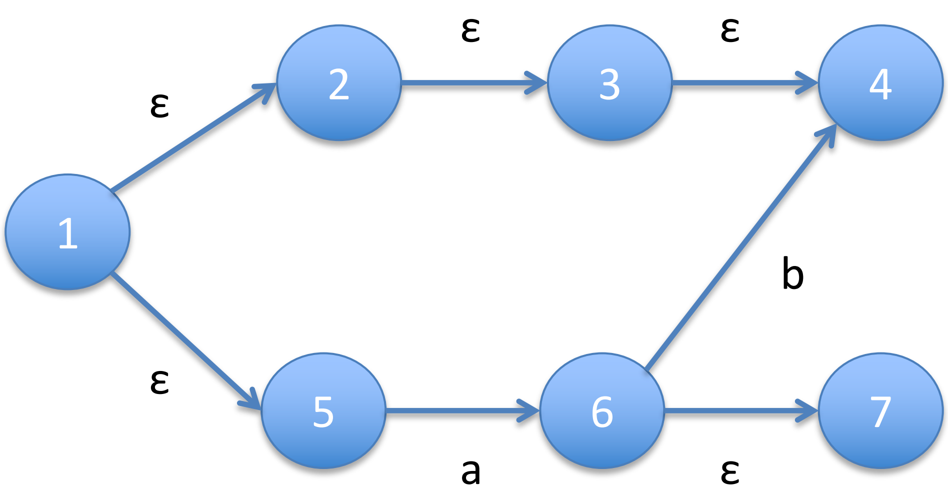
1. follow as many ε-transitions as you want
2. Read a
3. follow as many ε-transitions as you want

ECLOSE(q) - ε-closure

- q is in ECLOSE(q)
- If p is in ECLOSE(q) and r is in δ(p,ε) then r is in ECLOSE(q)



Example



q	ECLOSE(q)
1	
2	
3	
4	
5	
6	
7	

The language of the ε-NFA

By induction on the length of the string w:

- $\underline{\delta}(q,\varepsilon) = \text{ECLOSE}(q)$
- $\underline{\delta}(q,wa) = \underline{\text{ECLOSE}}(\delta(q_1,a) \cup \ldots \cup \delta(q_k,a))$

  where $\underline{\delta}(q,w) = \{q_1,\ldots,q_k\}$ and

  $\underline{\text{ECLOSE}}(\{q_1',\ldots,q_k'\})$

  $= \text{ECLOSE}(q_1') \cup \ldots \cup \text{ECLOSE}(q_k')$

The language of $A = (Q, \Sigma, \delta, q_0, F)$

$L(A) = \{w \mid \underline{\delta}(q_0,w) \text{ contains a state from } F\}$

Let's do it!

Exercise 2.5.1 (a,b) and 2.5.3 (b)

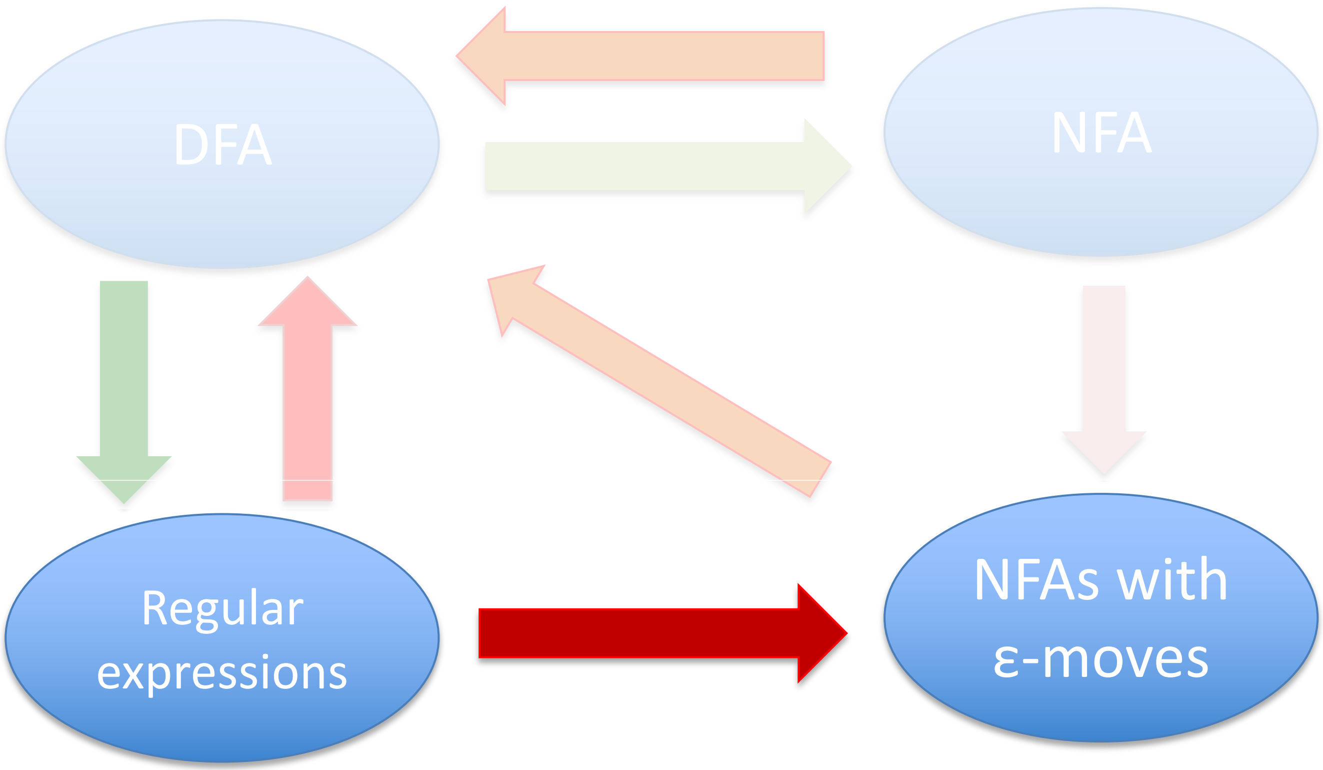
Exercise 2.5.1: Consider the following ε-NFA:

	ε	a	b	c
→p	∅	{p}	{q}	{r}
q	{p}	{q}	{r}	∅
*r	{q}	{r}	∅	{p}

a) compute the ε-closure of each state
b) give all the strings of length two or less accepted by the automaton

Exercise 2.5.3: Construct an ε-NFA accepting the set of strings either 01 repeated one or more times or 010 repeated one or more times.

The big picture



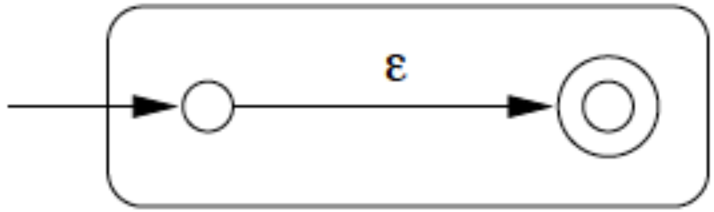
From RE to ε-NFA

- For each of the different forms of regular expressions we show how to construct an ε-NFA.
- We maintain the invariants:
  - Exactly one accepting state
  - No transitions entering the initial state
  - No transitions leaving the final state
- Recall what we did for Program Graphs

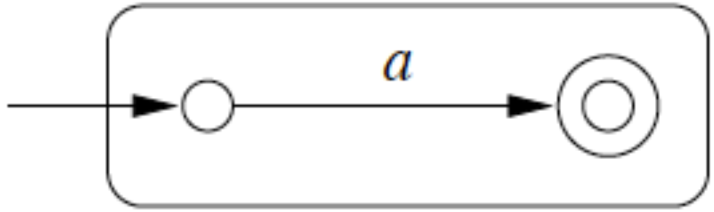
Automata for base cases

- ∅

- ε

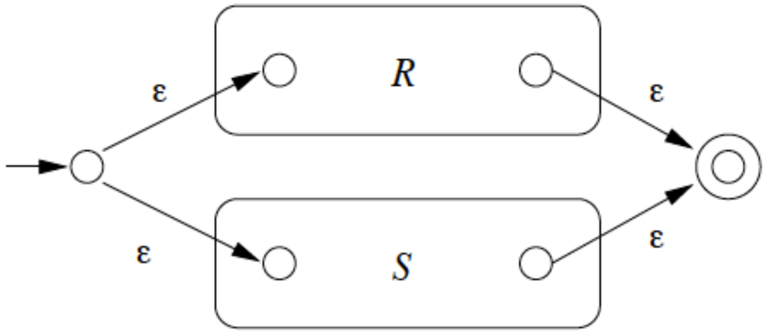


- a

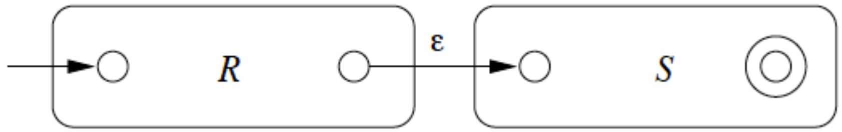


Automata in the inductive cases

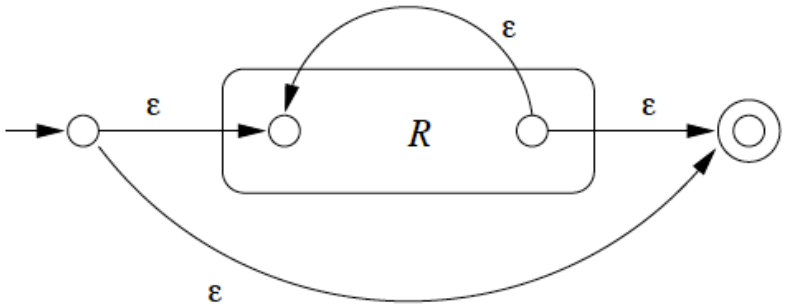
- R+S



- R S

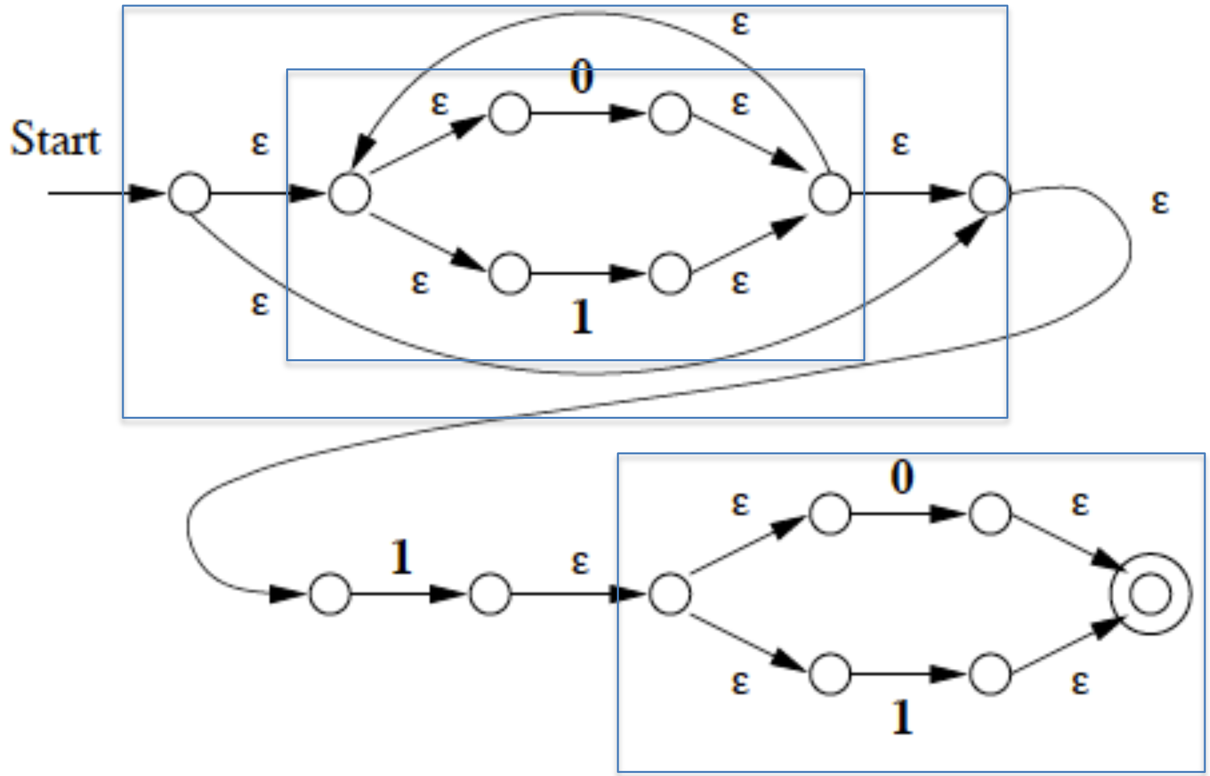


- R*



Complexity: O(n)

Example: (0 + 1)* 1 (0 + 1)



Let's do it!

Exercise 3.2.4(a) and 3.2.7

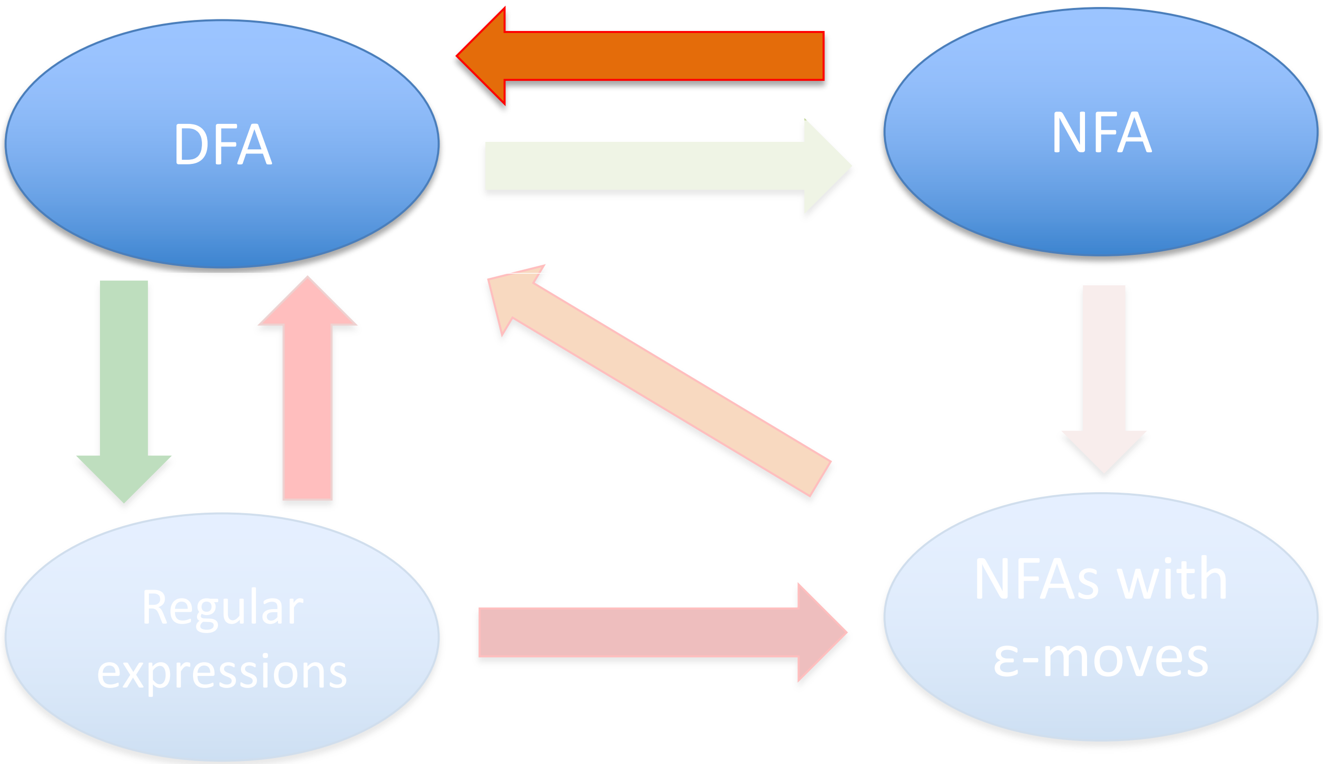
Exercise 3.2.4 (a) Use the algorithm to convert the regular expression 01* to an ε-NFA.

Exercise 3.2.7 Potential simplifications

1. For union: merge the two start states and the two accepting states rather than generating new ones
2. For concatenation: merge the accepting state of the first automata with the start state of the second
3. For closure: introduce ε-transitions between initial state and final state (rather than introducing the additional states)

Individually each of these simplifications are fine – but to what extend can they be combined?

The big picture



The subset construction

- For each NFA we construct a DFA that accepts the same language
  - The DFA may have 2^n states when the NFA have n states – exponential blow up!
  - But the good news is that in most cases it will have the same number of states but a larger number of transitions

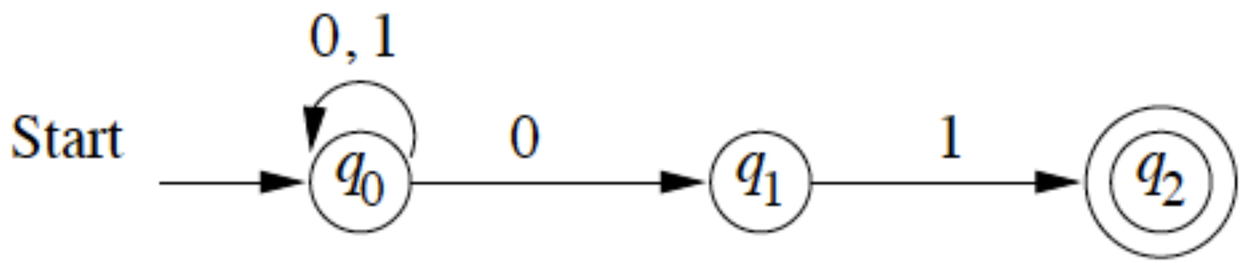
The subset construction

Complexity: $O(2^n)$

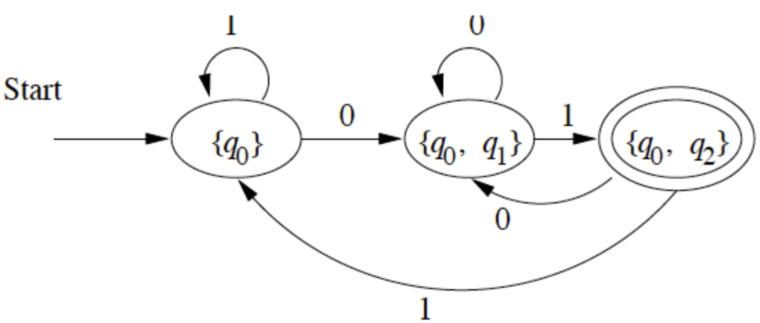
- Let $A_N = (Q_N, \Sigma, \delta_N, q_{N0}, F_N)$ be a NFA.
- Construct the DFA $A_D = (Q_D, \Sigma, \delta_D, q_{D0}, F_D)$ as follows:
  - Q_D is the set of all subsets of Q_N (so $Q_D = P(Q_N)$)
  - q_{D0} is the set $\{q_{F0}\}$
  - F_D is the subsets of Q_N that contains states from F_N
  - $\delta_D: Q_D \times \Sigma \rightarrow Q_D$ constructed from $\delta_N: Q_N \times \Sigma \rightarrow P(Q_N)$

$$\boxed{\delta_D(\{q_1,\ldots,q_k\},a) = \delta_N(q_1,a) \cup \ldots \cup \delta_N(q_k,a)}$$

Example



Becomes



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

all the subsets of $\{q_0,q_1,q_2\}$

Final states

Let’s do it!

Exercise 2.3.3

- Use the algorithm to convert the following NFA to a DFA and informally describe the language it accepts:

	0	1
→p	{p,q}	{p}
q	{r,s}	{t}
r	{p,r}	{t}
*s	Ø	Ø
*t	Ø	Ø

Is our construction correct?

- Let $A = (Q, \Sigma, \delta_N, q_0, F)$ is a NFA and assume that the DFA $A = (Q, \Sigma, \delta_D, q_0, F)$ is constructed by the subset construction.
- Then $L(A) = L(A)$

- Proof: It is sufficient to show that

$$\underline{\delta}_N(q,w) = \underline{\delta}_D(\{q\},w)$$

for all q and w

Proof

- To prove $\underline{\delta}_N(q,w) = \underline{\delta}_D(\{q\},w)$ we proceed by induction on the length of the string w

$\underline{\delta}_N(q,\varepsilon) = \{q\}$

$\underline{\delta}_N(q,wa) = \delta_N(q_1,a) \cup \ldots \cup \delta_N(q_k,a)$

where $\underline{\delta}_N(q,w) = \{q_1,\ldots,q_k\}$

NFA

$\underline{\delta}_D(\{q\},\varepsilon) = \{q\}$

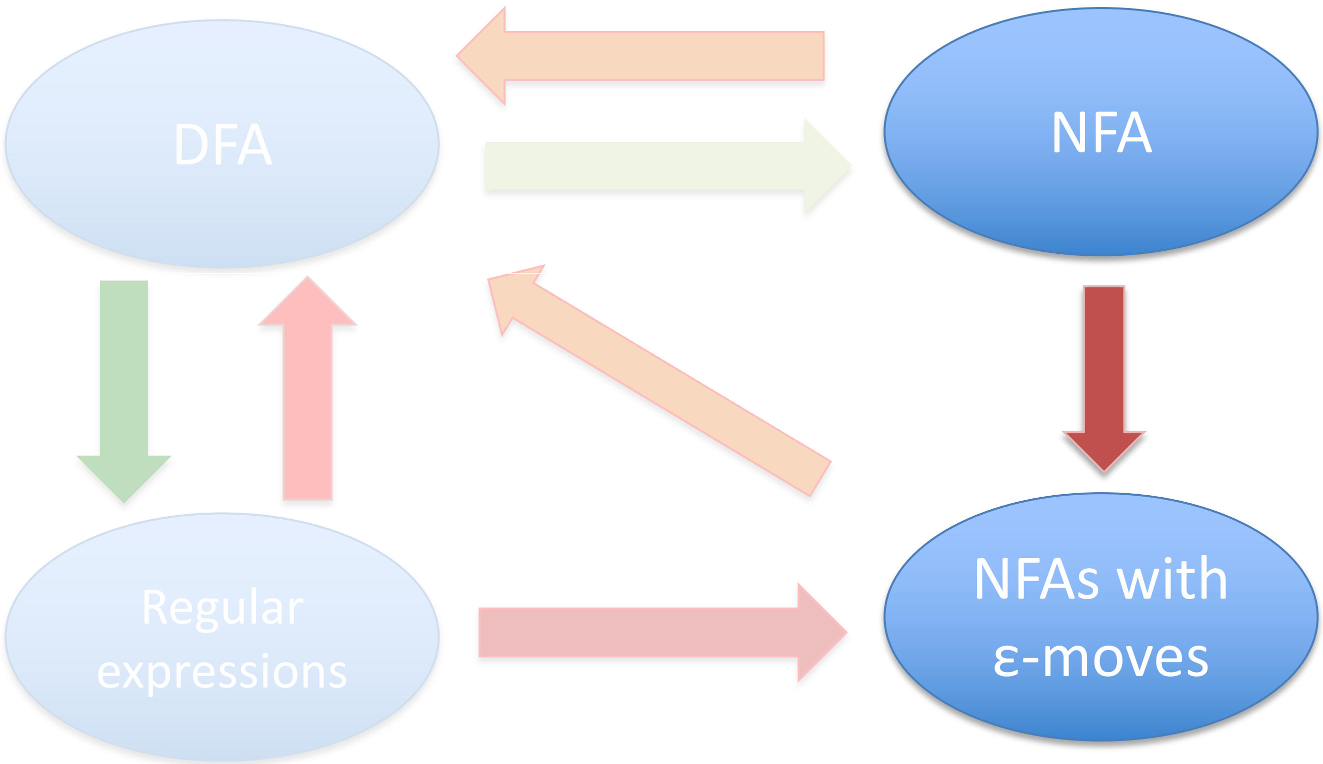
$\underline{\delta}_D(\{q\},wa) = \delta_D(\{q_1,\ldots,q_k\},a)$

where $\underline{\delta}_D(\{q\},w) = \{q_1,\ldots,q_k\}$

DFA

$$\boxed{\delta_N(q_1,a) \cup \ldots \cup \delta_N(q_k,a) = \delta_D(\{q_1,\ldots,q_k\},a)}$$

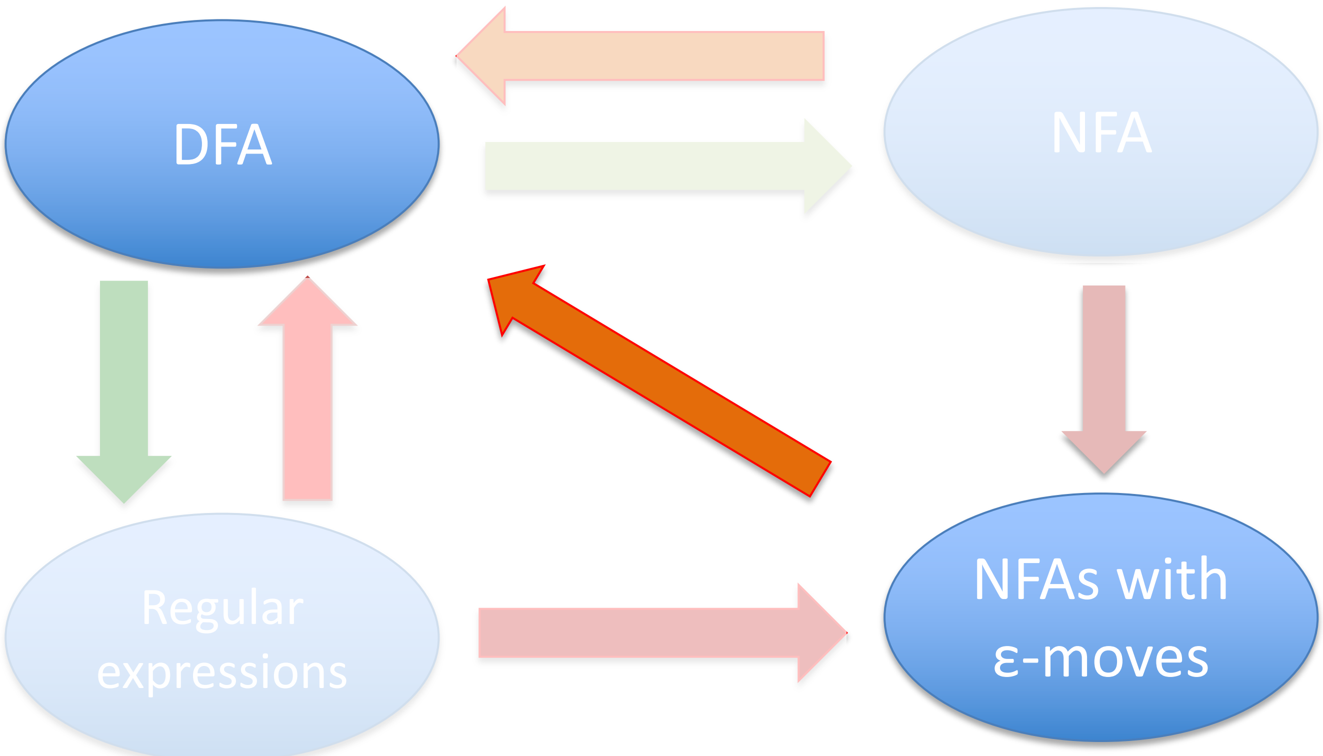
The big picture



From NFA to ε-NFA

- Any language accepted by an NFA is also accepted by an ε-NFA
  - This is trivial – but why?

The big picture



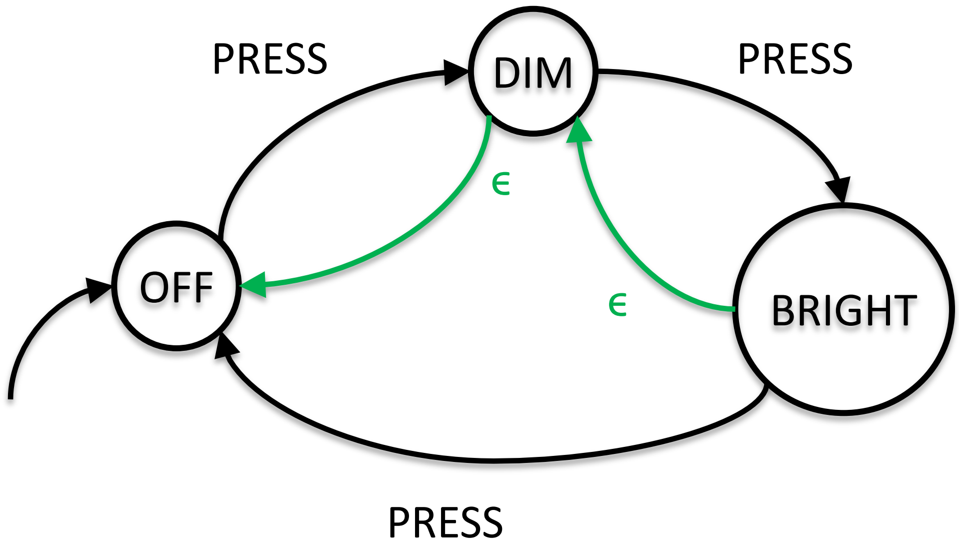
From ε-NFA to DFA

- We shall mimic the subset construction
- Given a ε-NFA $(Q_E, \Sigma, \delta_E, q_{E0}, F_E)$
- We shall construct a DFA $(Q_D, \Sigma, \delta_D, q_{D0}, F_D)$:
  - $Q_D = P(Q_E)$ – the subsets of Q_E
  - $q_{D0} = \text{ECLOSE}(q_{E0})$
  - F_D: all subsets of Q_E containing states from F_E
  - Transition function:

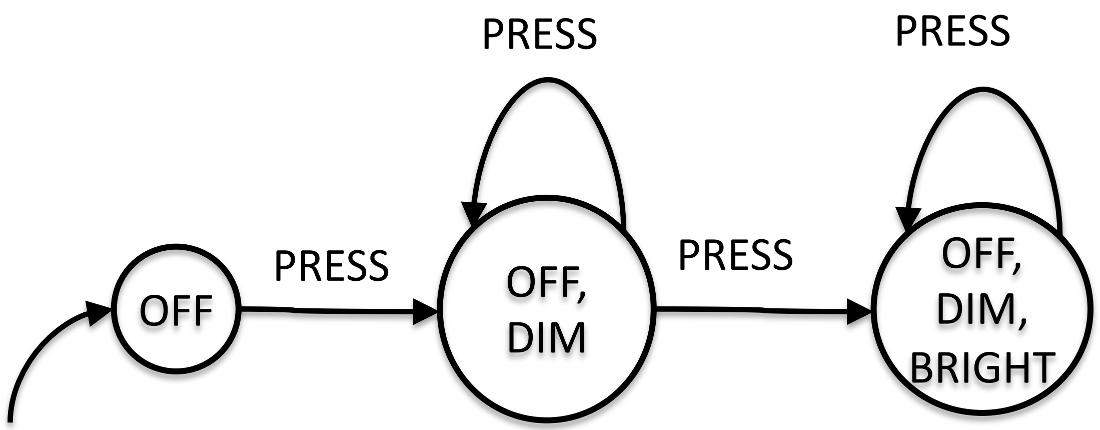
$$\delta_D(\{q_1,\ldots,q_k\},a) = \underline{\text{ECLOSE}}(\ \delta_E(q_1,a) \cup \ldots \cup \delta_E(q_k,a)\ )$$

Example

The ε-NFA



Is transformed into the DFA



From ε-NFA to DFA

- How can we prove that this is correct?

- We prove that for all strings w

  $\underline{\delta}_E(q_0,w) = \underline{\delta}_D(\mathrm{ECLOSE}(q_0),w)$

- The proof is by induction on w – see the book!

Let's do it!

Exercise 2.5.1 (again)

- Consider the following ε-NFA:

	ε	a	b	c
→p	Ø	{p}	{q}	{r}
q	{p}	{q}	{r}	Ø
*r	{q}	{r}	Ø	{p}

a) compute the ε-closure of each state

b) give all the strings of length two or less accepted by the automaton

c) convert the automaton to a DFA

READING MATERIAL AND EXERCISES

Reading material

- Covered in the lecture today:
  - HMU chapter 2: pages 60-65, 72-79
  - HMU chapter 3: pages 102-107
- Exercises for today
  - From NFAs to DFAs: HMU 2.3.3
  - On ε-NFAs: HMU 2.5.1, 2.5.3(b)
  - From RE to ε-NFAs: HMU 3.2.4(a,b), 3.2.5, 3.2.7
  - For 3.2.5: you can try two approaches (and compare the results):
    - constructing the ε-NFA with the algorithm, and then simplifying it.
    - constructing the DFA directly