Representations of Regular Languages

COMPSCI 3331

Outline

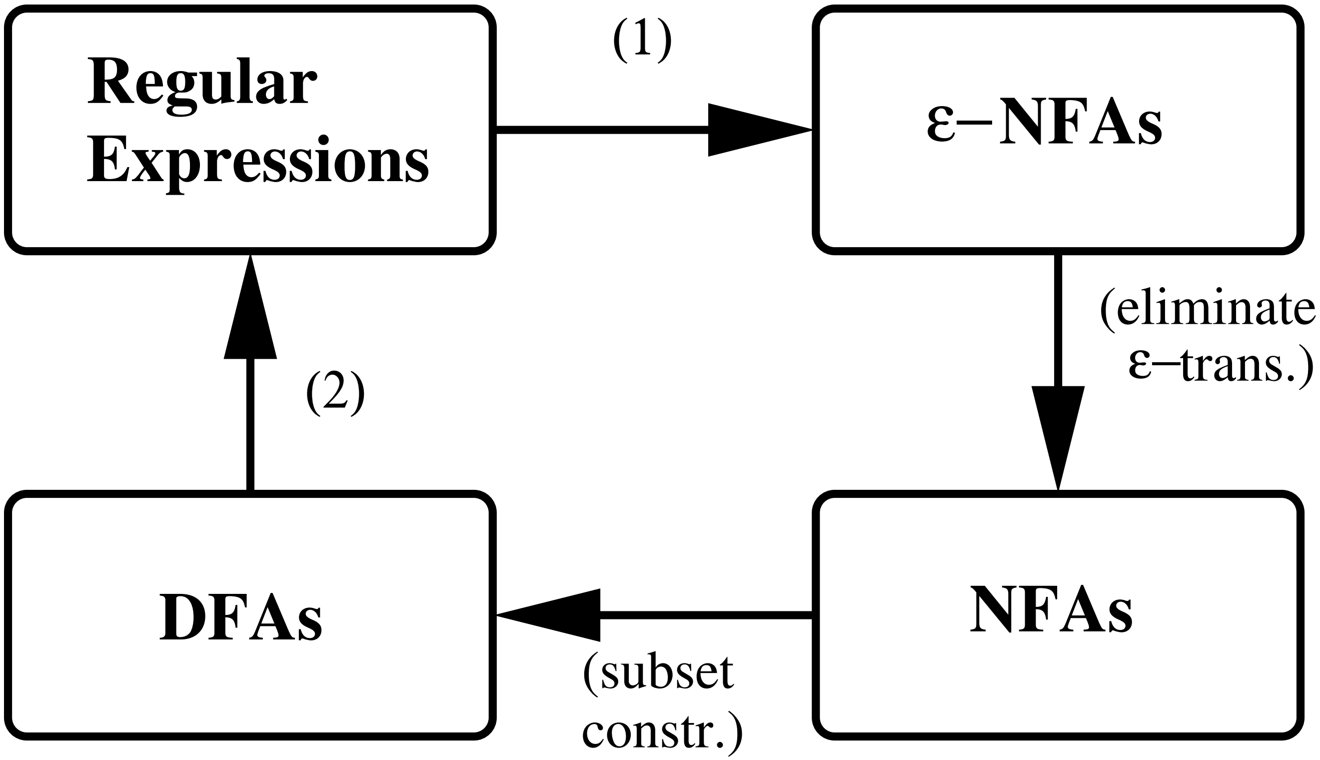
- Equivalence of Regular Expressions, NFAs, DFAs and ε-NFAs.
- Regular Expressions to ε-NFAs.
- DFAs to Regular Expressions.

What do we already know?

- Every DFA is an NFA.
- How to convert an NFA to a DFA
  - subset construction
  - can go from n to 2^n states.
- How to remove ε-transitions,
  - convert an ε-NFA to an NFA
  - number of states stays the same.

What about regular expressions?

Representations of Regular Languages



From RE to ε-NFA

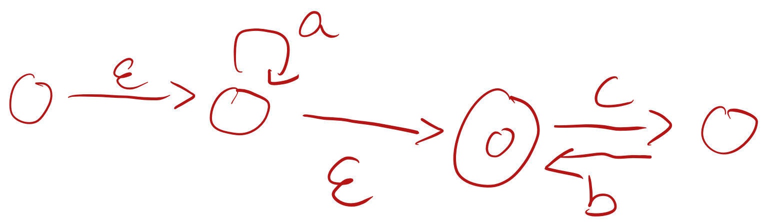
- **GOAL**: For every regular expression r define an ε-NFA M_r.
- Since regular expressions are defined recursively, the translation will be recursive.

Remember: Regular expressions are defined as:

- ε, $\emptyset$, a for all $a \in \Sigma$.
- $r_1 + r_2$, $r_1 r_2$ and r_1^* for any regular expressions r_1, r_2.

Intuitive conversion: RE to ε-NFA

Convert $a^*(cb)^*$ to an ε-NFA.



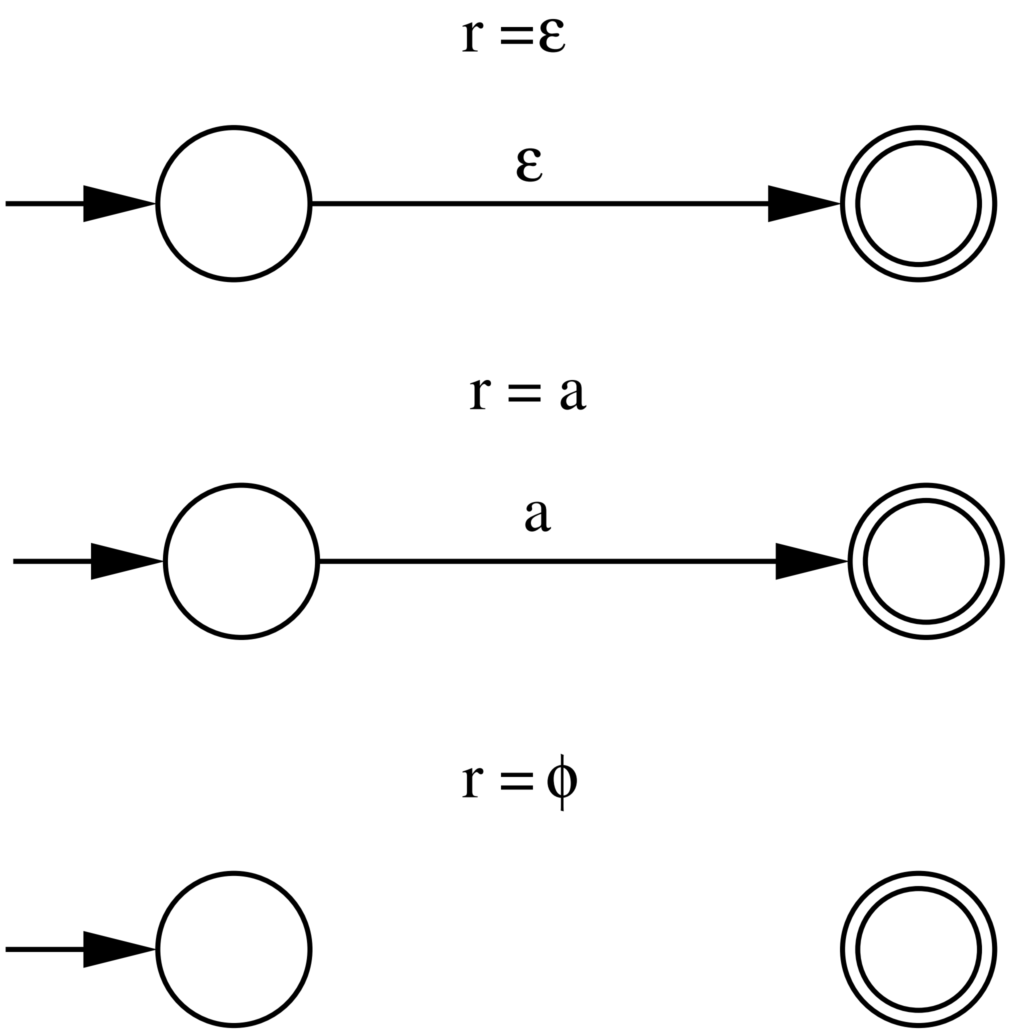
From RE to ε-NFA

We translate r to a ε-NFA M_r with:

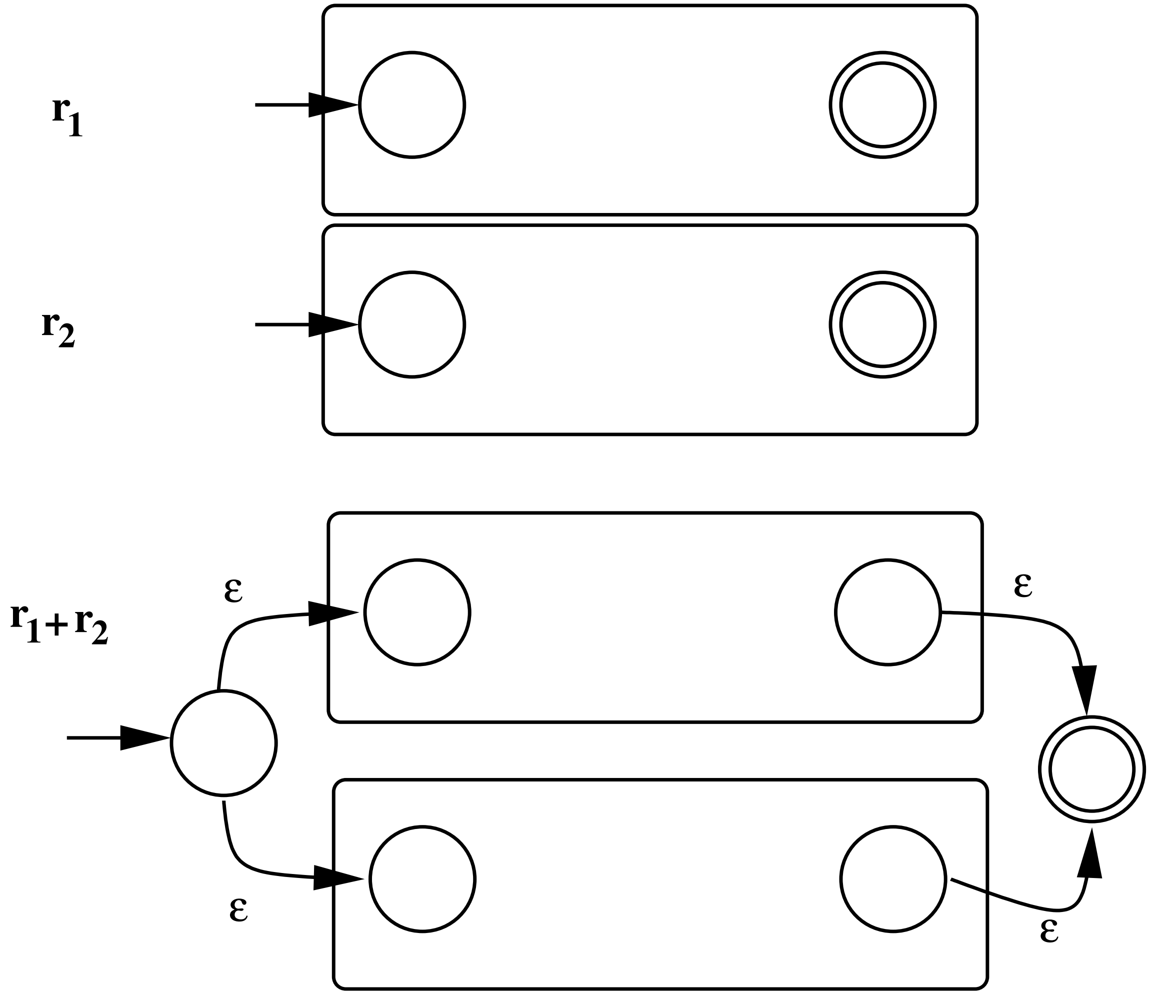
- only one final state.
- start state different from the final state.
- no transitions leaving the only final state.
- no transitions entering the initial state.

From RE to ε-NFA: Base Cases

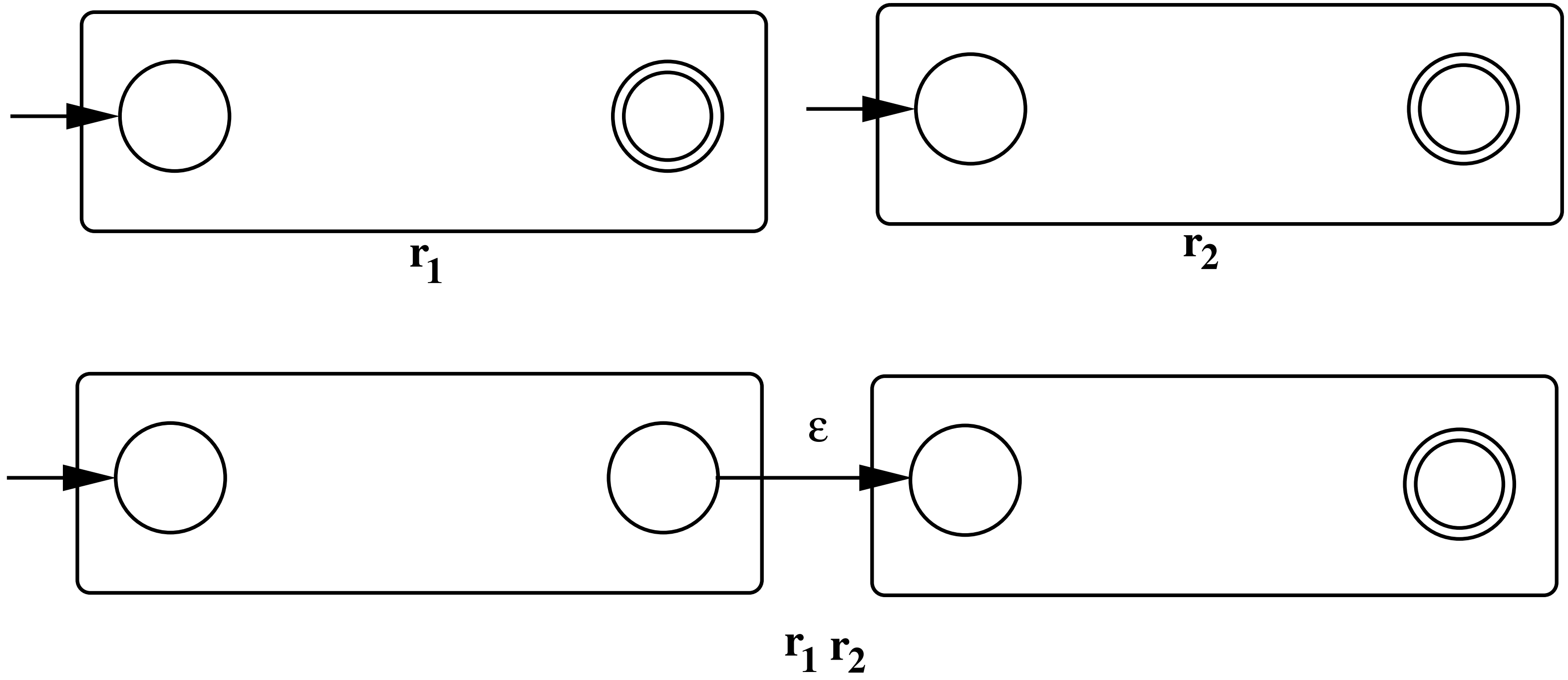
Base cases: $\emptyset$, ε and a for all $a \in \Sigma$.



From RE to ε-NFA: Union

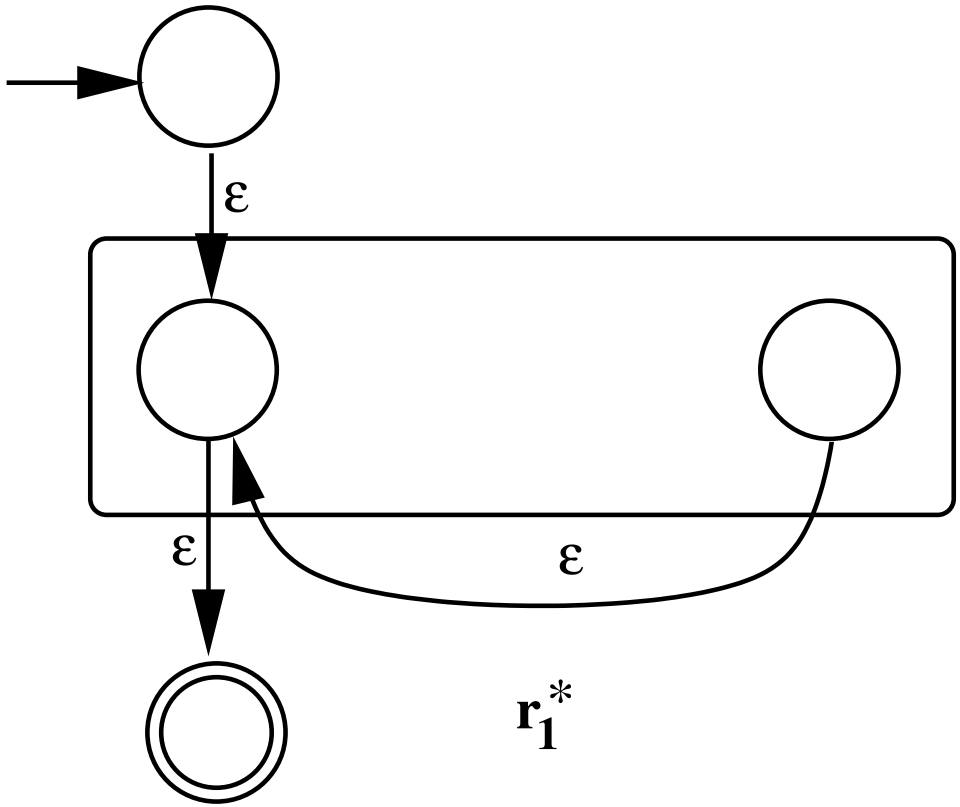


From RE to ε-NFA: Concatenation



From RE to ε-NFA: Kleene Star

$\mathbf{r_1}$



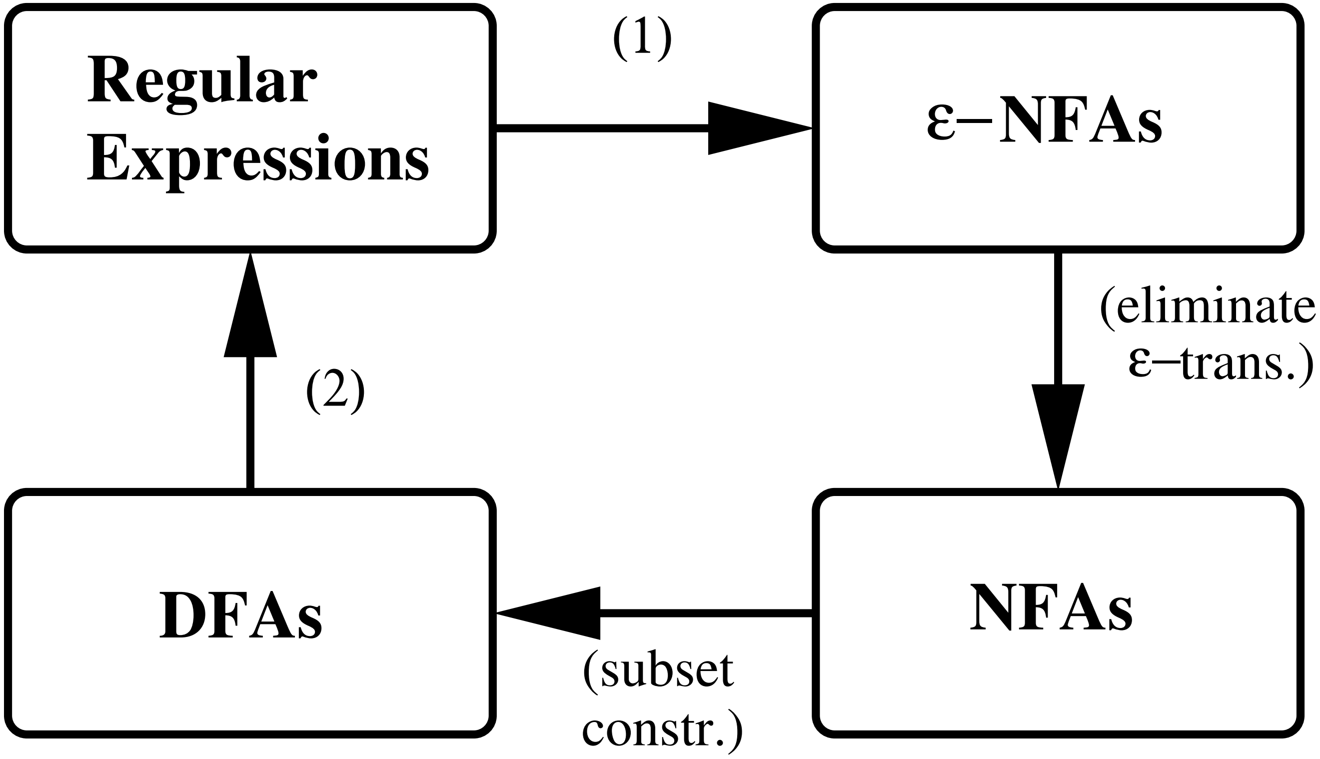
$\mathbf{r_1^*}$

From RE to ε-NFA

Theorem. Let r be a regular expression and let M_r be the ε-NFA we get from applying the previous rules. Then $L(r) = L(M_r)$.

Proof. By structural induction.

Conversion



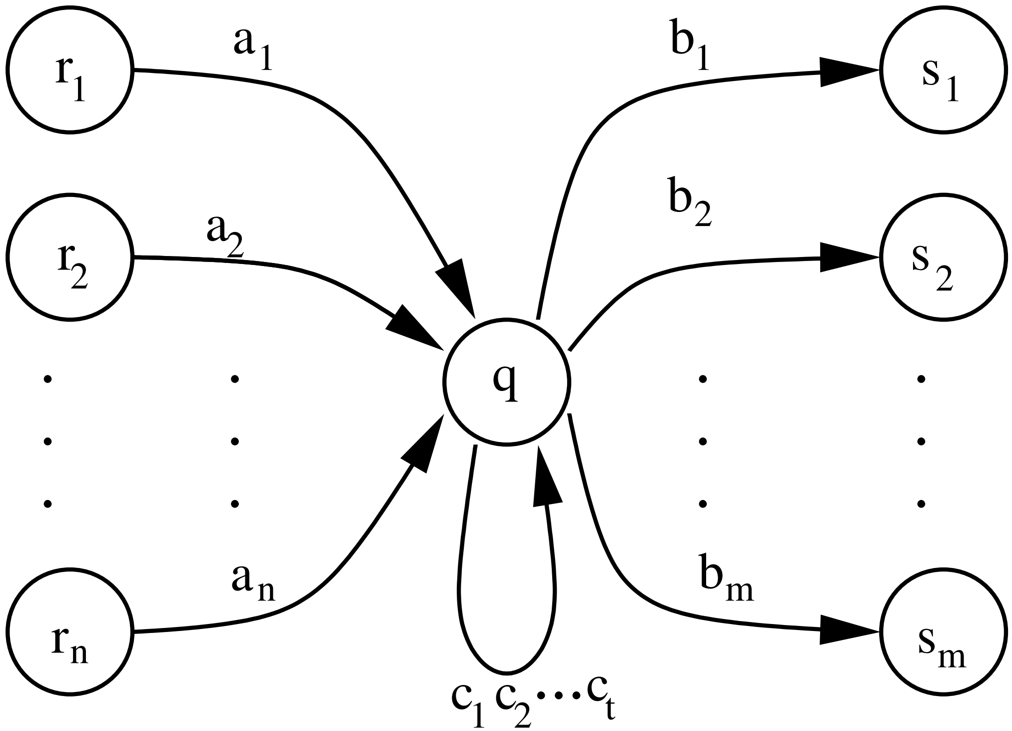
Regular Expressions
(1)
ε–NFAs
(eliminate ε–trans.)
NFAs
(subset constr.)
DFAs
(2)

From DFA to RE (1 of 7)

- MAIN IDEA: Paths through a DFA correspond to parts of a regular expression for that DFA.
- State Elimination method: eliminate states one-by-one until we get a regular expression for the entire machine.

From DFA to RE (2 of 7)

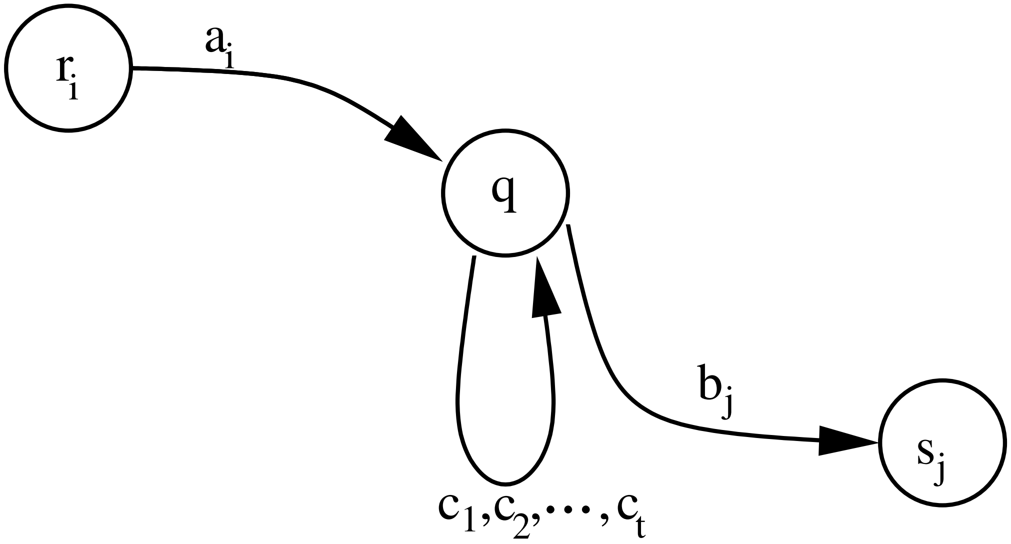
Consider an arbitrary state q of a DFA. What's going on at q?



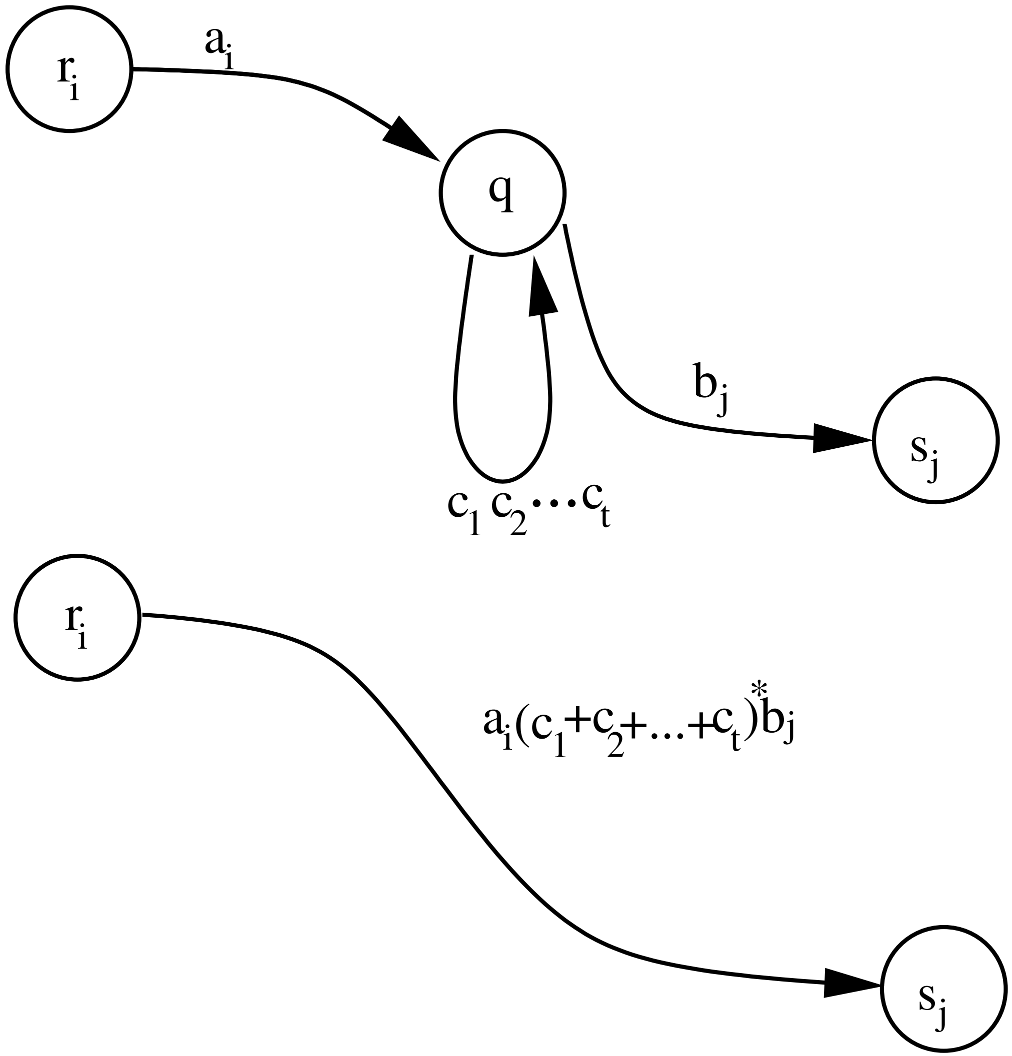
What words take us from r_i to s_j (for arbitrary $1 \leq i \leq n$ and $1 \leq j \leq m$)?

From DFA to RE (3 of 7)

Focus on one of the r_i and one of the s_j:



MAIN IDEA: we can replace the transitions with one transition labelled $a_i(c_1 + \cdots + c_t)^* b_j$.

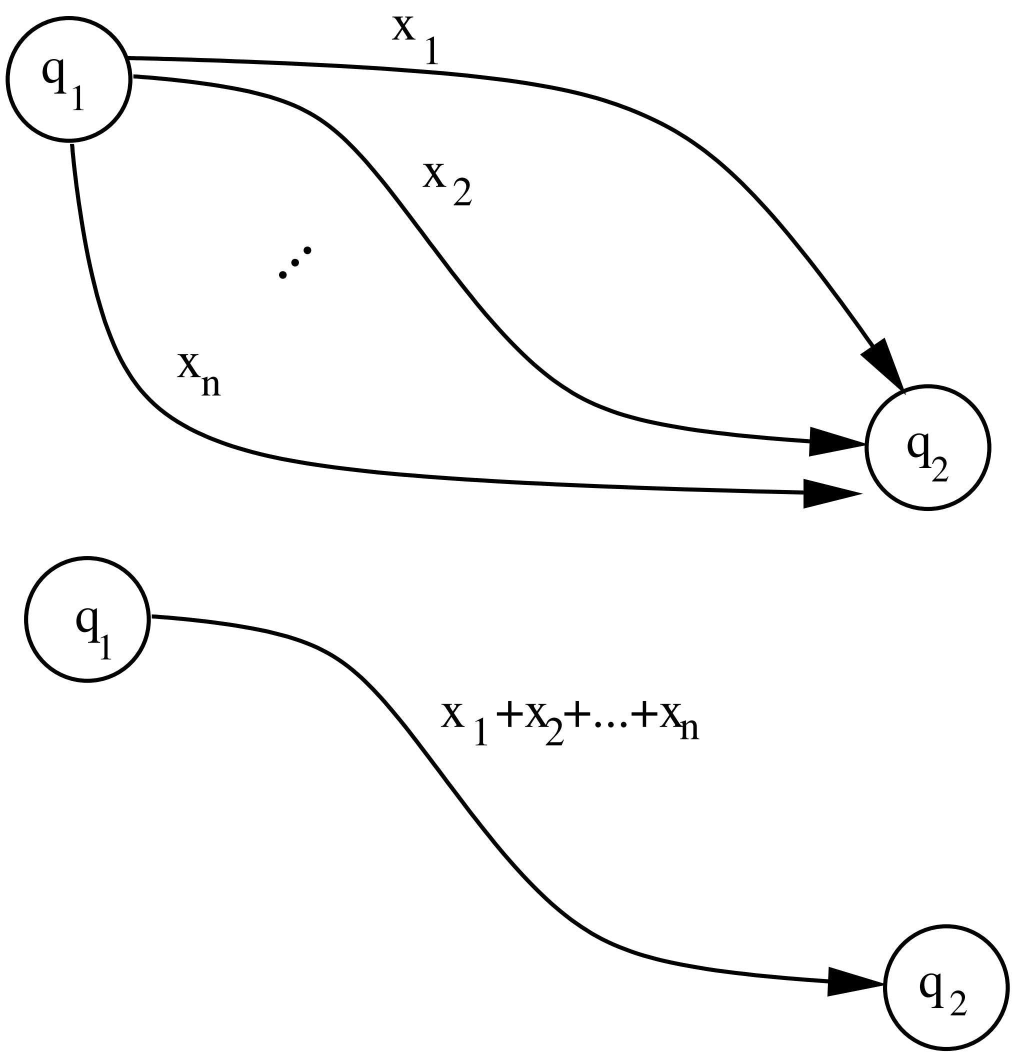


From DFA to RE (5 of 7)

- Everything in the previous argument applies to **regular expressions** on transitions and not just letters.
- Apply the replacement to each (incoming, outgoing) transition pair before removing a state.
- Repeatedly apply removal process to states until we get a regular expression for the DFA.

From DFA to RE (6 of 7)

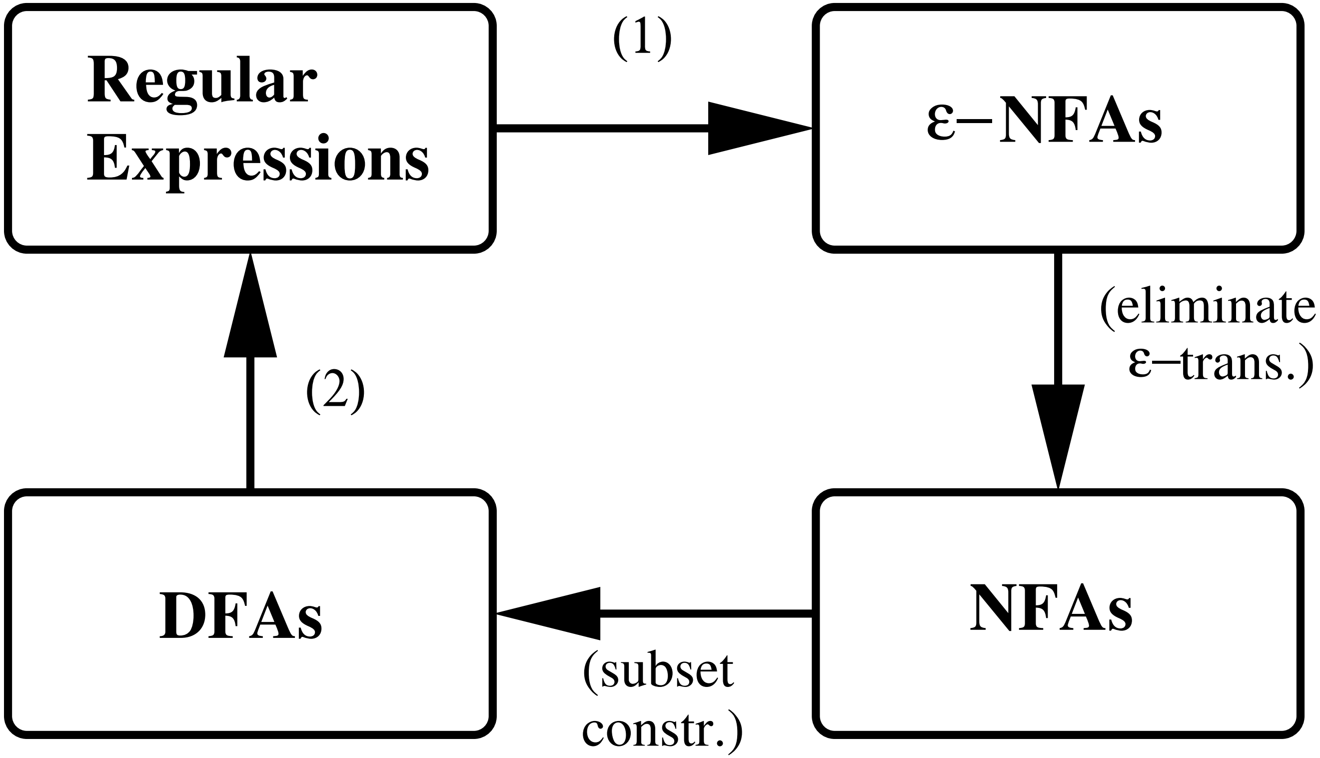
Replace multiple transitions from state q_1 to q_2 with one regular expression.



From DFA to RE (7 of 7)

- Some pre-processing before beginning:
  - add a new start state and connect it to the old start state by an ε-transition
  - add a new final state and make all old final states point to it with an ε-transition
- Gives us states that won't be eliminated during the process.
- Eliminate any state which is not initial or final. At the end, read the regular expression for the DFA.

Representations Summary



(1) recursive translation of regular expressions to ε-NFAs.

(2) state elimination method.

Representations of Regular Languages: Summary

Theorem. For every regular expression r there exists a DFA M such that $L(r) = L(M)$. For every DFA M, there exists a regular expression r such that $L(M) = L(r)$.