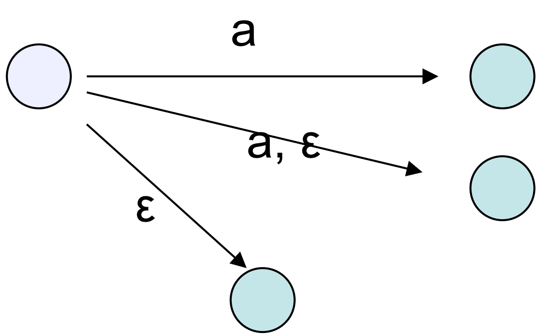
NFA with ε moves

- Let ε denote the null string.
- Extend the transition function $f: Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow P(Q)$
  - E.g., f(a) = {q1, q2}; f(ε) = {q2, q3}

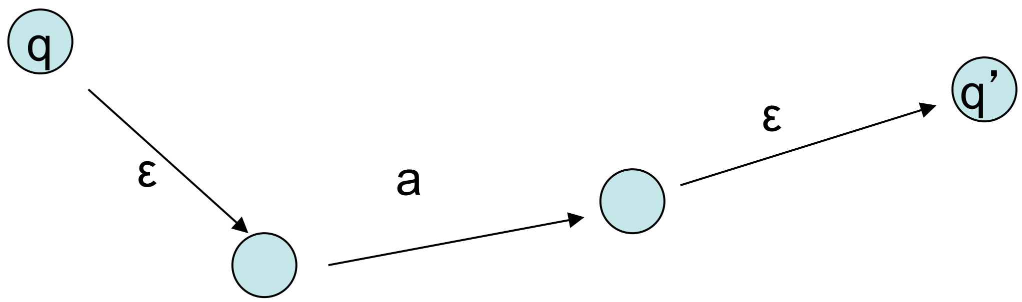


- Are NFA with ε moves more powerful than NFA and DNA? No.

Lemma. Given an NFA M_ε with ε moves, we can construct another NFA M without ε moves accepting the same language.

Idea. M uses the same set of states as M_ε.

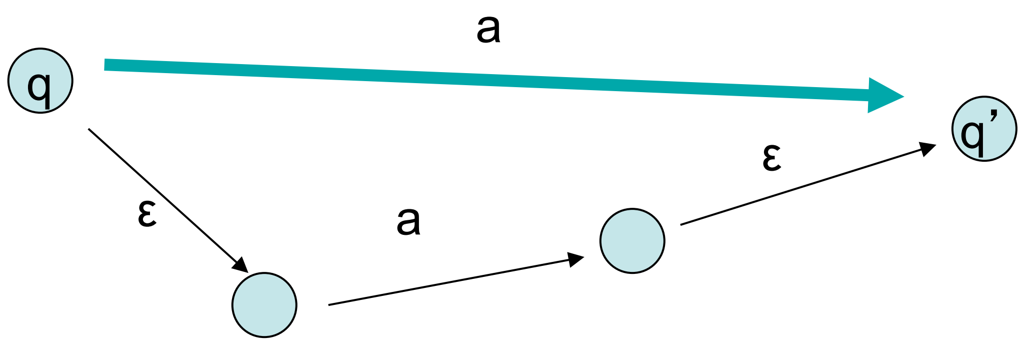
For every pair of states (q, q'),
M has a transition from q to q' labeled with a in Σ
if and only if
in M_ε, there is a path from q to q' labeled with all ε except one a.



Lemma. Given an NFA M_ε with ε moves, we can construct another NFA M without ε moves accepting the same language.

Idea. M uses the same set of states as M_ε.

For every pair of states (q, q'),
M has a transition from q to q' labeled with a in ∑
if and only if
in M_ε, there is a path from q to q' labeled with all ε except one a.



Week 1: finite automata

- Summary: Finite automata, deterministic versus nondeterministic computation, languages, limitation of finite automata, regular expressions.
- Reading:
  - Sipser, Chapter 1 (and Chapter 0 for those not confident in discrete math); or
  - Hopcroft, Motwani & Ullman (1st edition or 2nd edition), Chapter 2 & 3.2

Non-computational models

Given a language (decision problem) L, we can reason whether there is a finite automaton (pda, or Turing machine) accepting L.

In fact, languages can be characterised by some non-computation-based models.

- Regular expressions — Today's lecture
  - Theorem. A language L is accepted by a DFA if and only if L = L(R) of some regular expression R.
- Right (Left) Linear grammars — Probably exercise
  - Theorem. A language L is accepted by a DFA if and only if L = L(G) of some right linear grammar R.

Equivalence

- DFA
- NFA
- NFA with ε moves
- Regular expressions

Regular Expressions

- A simple way to define a set of strings (a language).
- For example, (0 U 1)0* denotes the set { 0, 1, 00, 10, 000, 100, 0000, 1000, …}
- A recursive definition: R is a regular expression over an alphabet Σ if R is
  - *a* for some *a* in **Σ**, ε, ∅,
  - (R_1 U R_2), (R_1 o R_2), or R_1*, where R_1 and R_2 are regular expressions.

$R_1 R_2$

- More examples: 0*10*, (0U1)*1

Languages

A regular expression R defines a language **L**(R).

- R = a: L(R) = {a}.
- R = ε: L(R) = {ε} (i.e., the set of null string).
- R = $\emptyset$: L(R) is empty.
- R = (R_1 U R_2): L(R) = { w | w is in L(R_1) or L(R_2) }.
- R = (R_1 o R_2): L(R) = { w | w = **xy** where **x** is in L(R_1) and **y** is in L(R_2) }.
- R = R_1*: L(R) = { w | w is in L(R_1)*}.

Def. w = ε or $w_1 w_2 w_3 \ldots w_n$, where $n \geq 1$ and each w_i is in L(R_1).

Examples

- $(01^*)^*$:

For convenience: we let R^+ be shorthand for RR^*.

- $(01^+)^*$:
- $1^* \varnothing$:
- ε^*

Regular Expressions & DFA

Theorem. Let L be a language accepted by a DFA M. Then there exists a regular expression R such that L(R) = L.

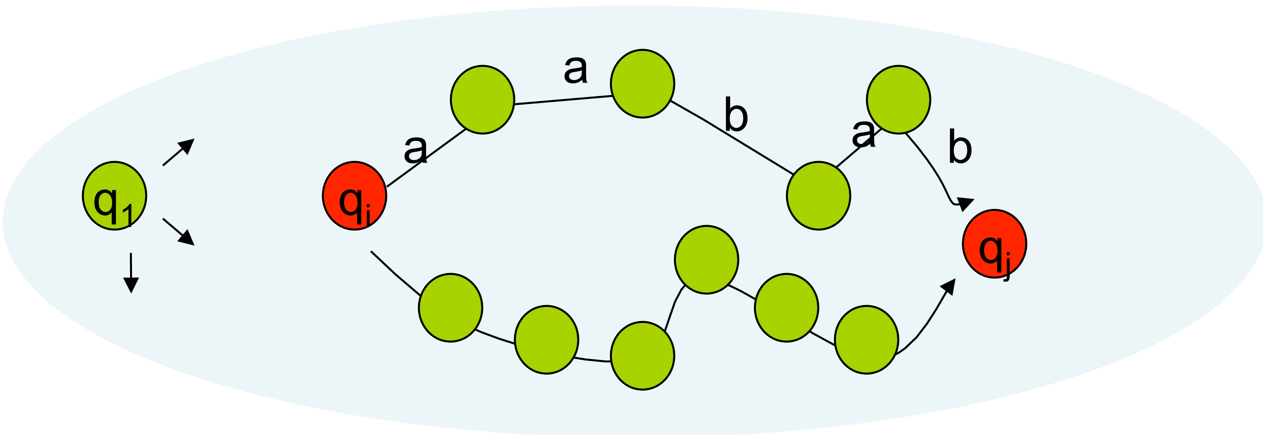
Implication: An NFA or DFA is no more powerful than a regular expression.

Regular Expressions & DFA

Theorem. If L is accepted by a DFA M, then L = L(R) for some regular expression R.

Suppose that M = (Q, Σ, f, q_1, F) and Q = {q_1, q_2, ..., q_n}, where n denotes the number of states.

The transition function of M (i.e., f) defines a directed graph, in which every vertex is a state and every edge is labeled with a symbol in Σ.

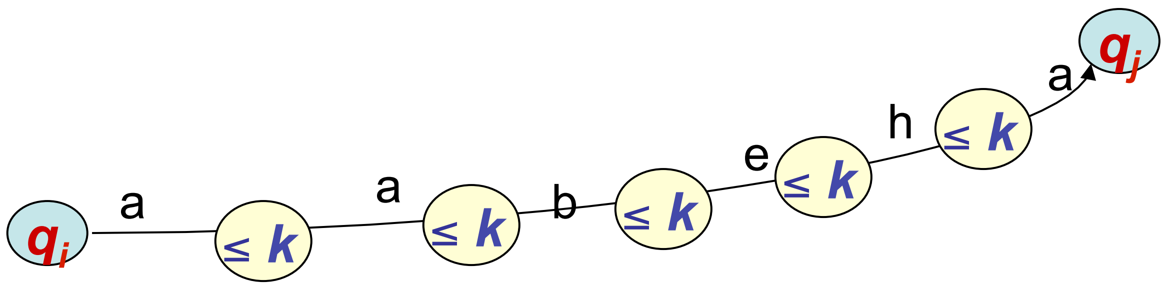


From State q_i to State q_j

Consider any k in [0, n], and i,j in [1,n].

Let $S_{i,j}$ = { x | x is the string on a path from q_i to q_j in M }

Let $S_{i,j}(k)$ = { x | x is the string on a path from q_i to q_j in M, and excluding the two ends, every state on this path has a label q_b with $b \le k$. }

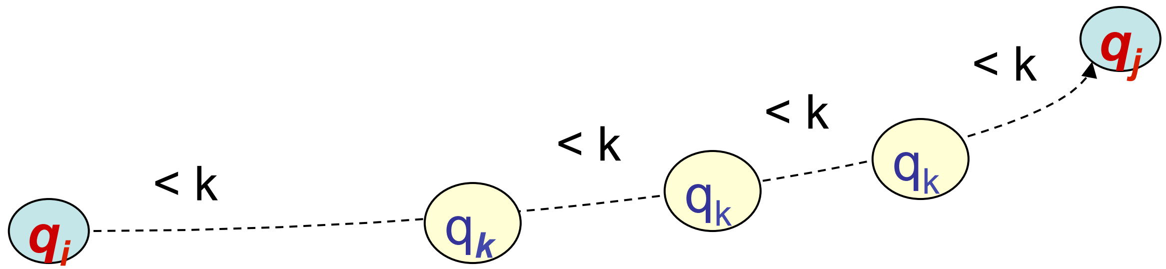


NB. $S_{i,j}(k)$ is a set, not a regular expression.

<u>Question</u>: What is $S_{i,j}(0)$?

A Technical Lemma

Lemma. $S_{i,j}(k) = S_{i,j}(k\text{-}1) \cup S_{ik}(k\text{-}1)\,(S_{kk}(k\text{-}1))^*\,S_{kj}(k\text{-}1)$.



$S_{1,j}(n)$

Suppose q_j is a state in F.

What does $S_{1,j}(n) = S_{1,j}$ denote ?

$S_{1,j}(n)$

Suppose q_j is a state in F.

What does $S_{1,j}(n) = S_{1,j}$ denote ?

- The set of strings that M accepts using the final state q_j.

Let L be the language accepted by M.

Then L is equal to the *union* of all $S_{1,j}(n)$, where q_j is in F.

Converting DFA to regular expressions

Lemma For any i, j, k, there is a regular expression R such that $L(R) = S_{i,j}(k)$.

Proof. By induction on k.

Basis: $k = 0$.

Let $a_1, a_2, \ldots, a_h$ be symbols in Σ such that $f(q_i, a_1) = q_j$, $f(q_i, a_2) = q_j, \ldots, f(q_i, a_h) = q_j$.

Let R be the regular expression $a_1 \cup a_2 \cup \ldots \cup a_h$.

Then $L(R) = \{a_1, a_2, \ldots, a_h\} = S_{i,j}(0)$.

NB. If no such a exists, then $R = \emptyset$.

Induction Step

Assume that the lemma is true for k-1.

Recall that $S_{i,j}(k) = S_{i,j}(k-1)\ \cup\ S_{ik}(k-1)\ (S_{kk}(k-1))^*\ S_{kj}(k-1)$.

By <u>induction hypothesis</u>, there exist regular expressions R1, R2, R3, and R4 such that

- $L(R1) = S_{i,j}(k-1)$,
- $L(R2) = S_{ik}(k-1)$,
- $L(R3) = S_{kk}(k-1)$,
- $L(R4) = S_{kj}(k-1)$.

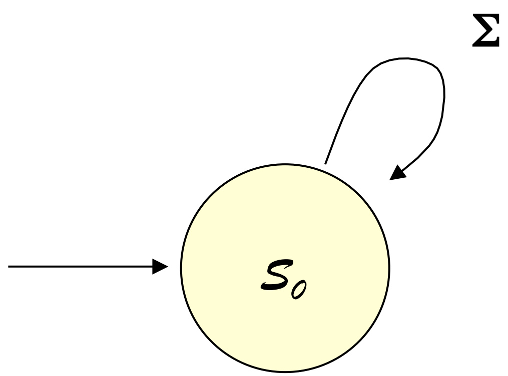
Then $R = R1 \cup (R2\ (R3)^*\ R4)$ is a regular expression such that $L(R) = S_{i,j}(k)$.

From regular Expressions to NFA

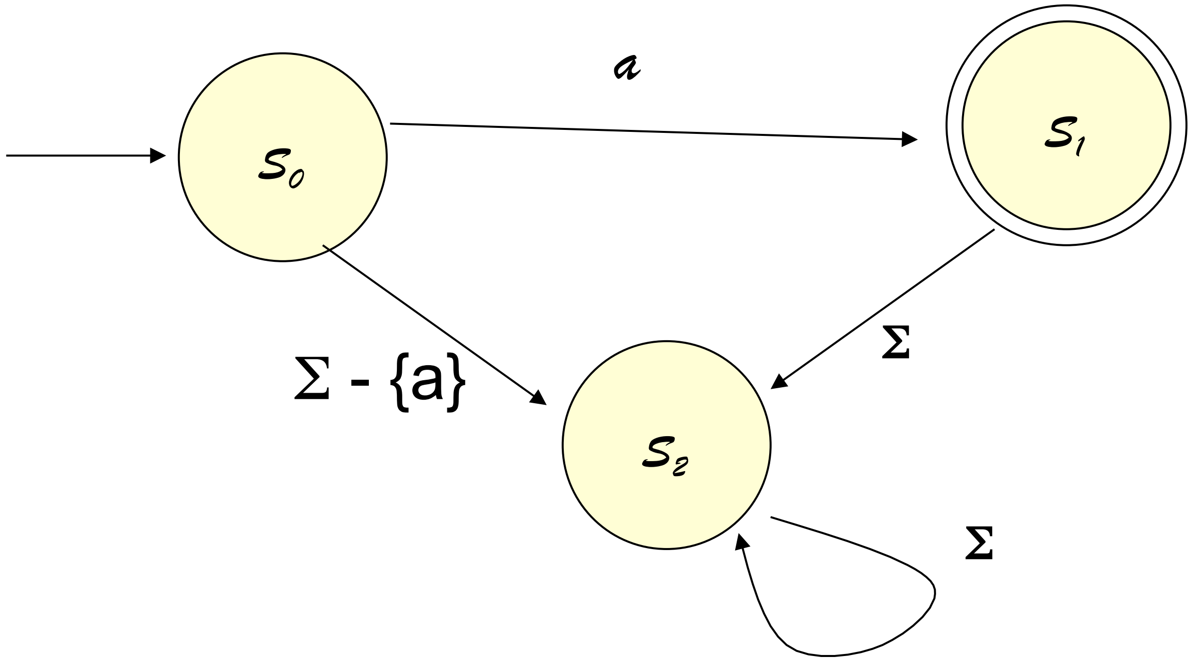
Theorem. Let r be a regular expression. Then there exists an NFA with ε moves M such that L(M) = L(R).

Proof. By induction on the structure of R.

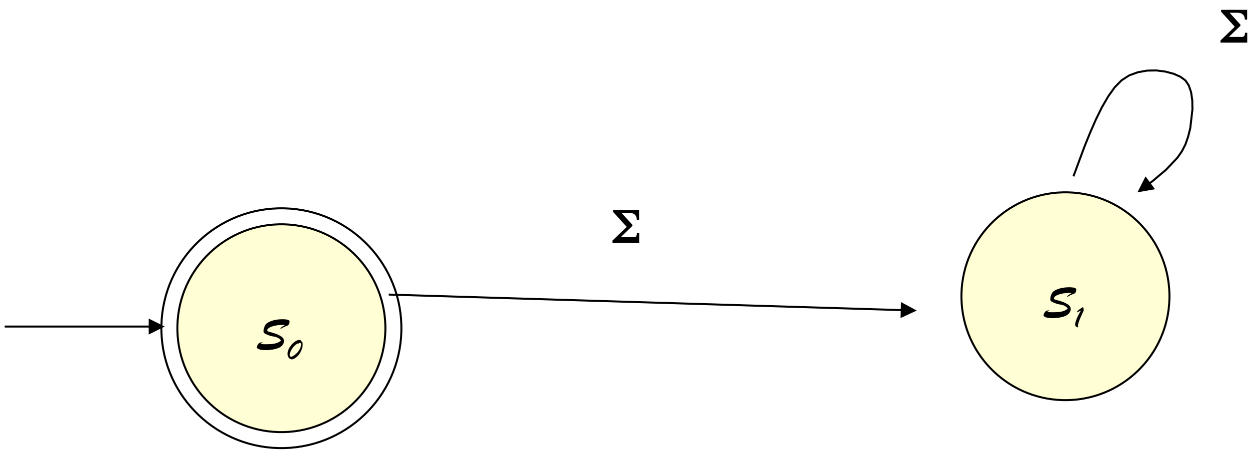
- R = ∅:



- R = a



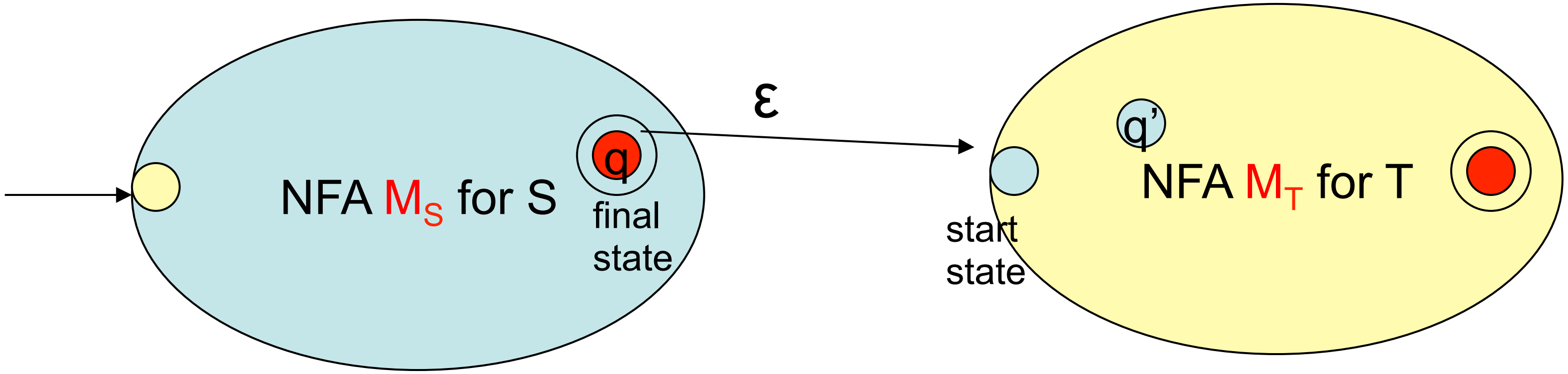
- R = ε



Induction Step

Consider a regular expression R. Assume the theorem is true for all sub-expressions of R.

Case 1. R= S T



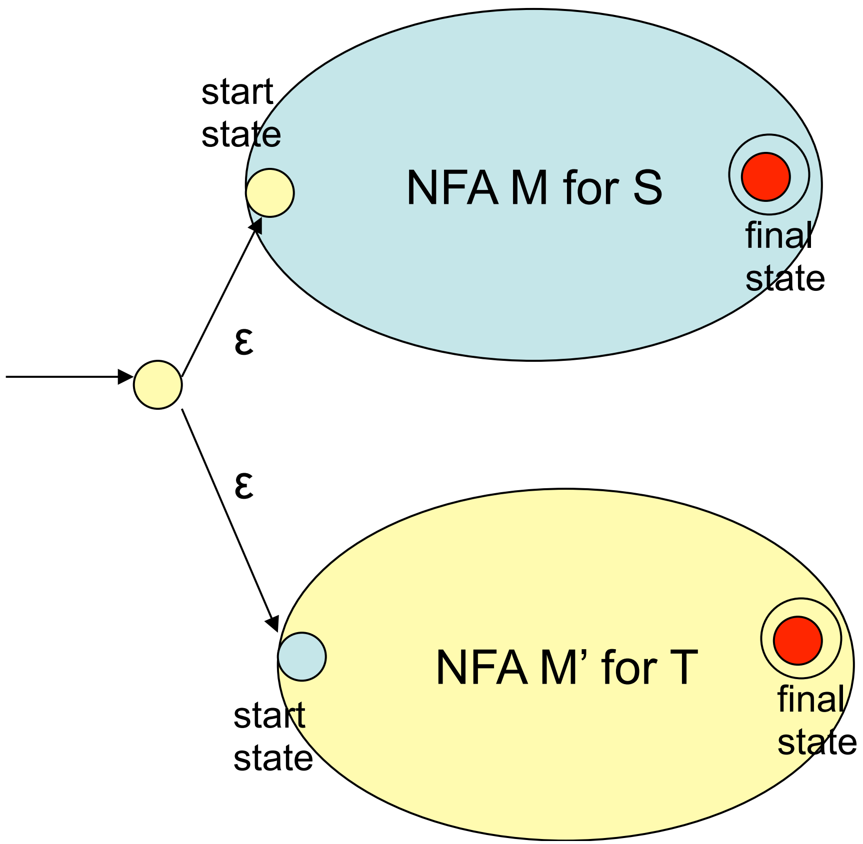
Combine the two NFA to make a bigger NFA for R.

- For each final state q of M_S, add an ε transition to the start state of M_T.
- New final states: All final states of M_T remain final states.

  What about the final states of M_S?

Case 2. R = S U T

Create a new start state, which has a ε transition to the start states of M_S and M_T.



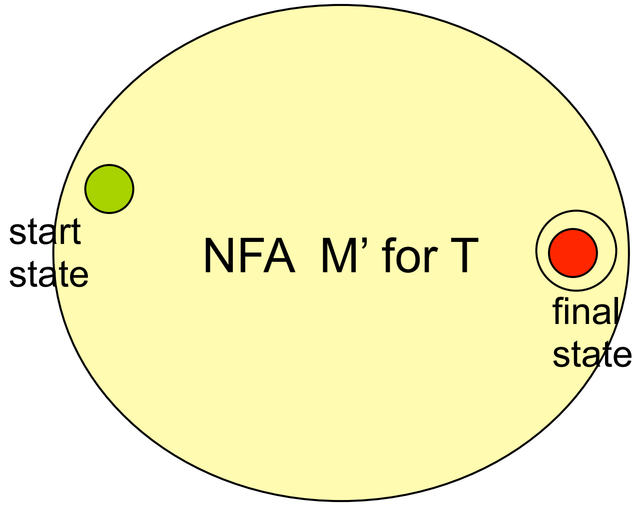
What are the new final states?

Final states of M_S and M_T.

Case 3. R = T*

Create a new start state, which is also a new final state, and has an ε move to the original start state.

Each original final state has an ε transition to the original start state.



Case 3. R = T*

Create a new start state, which is also a new final state, and has an ε move to the original start state.

Each original final state has an ε transition to the original start state.