Characterizing FA languages

- Regular expressions

Regular Expressions (Def. 1.52)

Given an alphabet Σ, R is a regular expression if:
(INDUCTIVE DEFINITION)

- R = a, with $a \in \Sigma$
- R = ε
- R = $\emptyset$
- R = $(R_1 \cup R_2)$, with R_1 and R_2 regular expressions
- R = $(R_1 \bullet R_2)$, with R_1 and R_2 regular expressions
- R = $(R_1{}^*)$, with R_1 a regular expression

Precedence order: *, $\bullet$, $\cup$

Regular Expressions

- Unix ‘grep’ command: Global Regular Expression and Print
- Lexical Analyzer Generators (part of compilers)
- Both use regular expression to DFA conversion

Examples

- $e_1 = a \cup b$, $L(e_1) = \{a,b\}$
- $e_2 = ab \cup ba$, $L(e_2) = \{ab,ba\}$
- $e_3 = a^*$, $L(e_3) = \{a\}^*$
- $e_4 = (a \cup b)^*$, $L(e_4) = \{a,b\}^*$
- $e_5 = (e_m \, . \, e_n)$, $L(e_5) = L(e_m) \bullet L(e_n)$
- $e_6 = a^*b \cup a^*bb$,

  $L(e_6) = \{w \mid w \in \{a,b\}^*$ and w has 0 or more a's followed by 1 or 2 b's$\}$

Characterizing Regular Expressions

• We prove that Regular expressions (RE) and Regular Languages are the same set, i.e.,

RE = RL

Thm 1.54: RL ~ RE

We need to prove both ways:

• If a language is described by a regular expression, then it is regular (Lemma 1.55)
(We will show we can convert a regular expression R into an NFA M such that L(R)=L(M))

• The second part:
If a language is regular, then it can be described by a regular expression (Lemma 1.60)

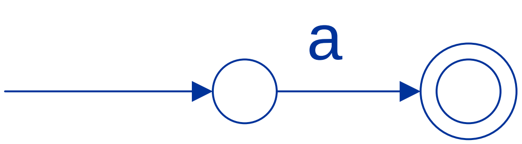
Regular expression to NFA

Claim: If L = L(e) for some RE *e*, then L = L(M) for some NFA M

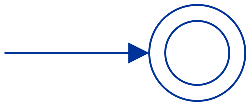
Construction: Use inductive definition

1. R = a, with $a \in \Sigma$
2. R = ε
3. R = $\emptyset$
4. R = $(R_1 \cup R_2)$, with R_1 and R_2 regular expressions
5. R = $(R_1 \bullet R_2)$, with R_1 and R_2 regular expressions
6. R = $(R_1{}^*)$, with R_1 a regular expression

1.



2.



3.

4,5,6: similar to closure of RL under regular operations.

Examples of RE to NFA conv.

L = {ab,ba}

L = {ab,abab,ababab,……}

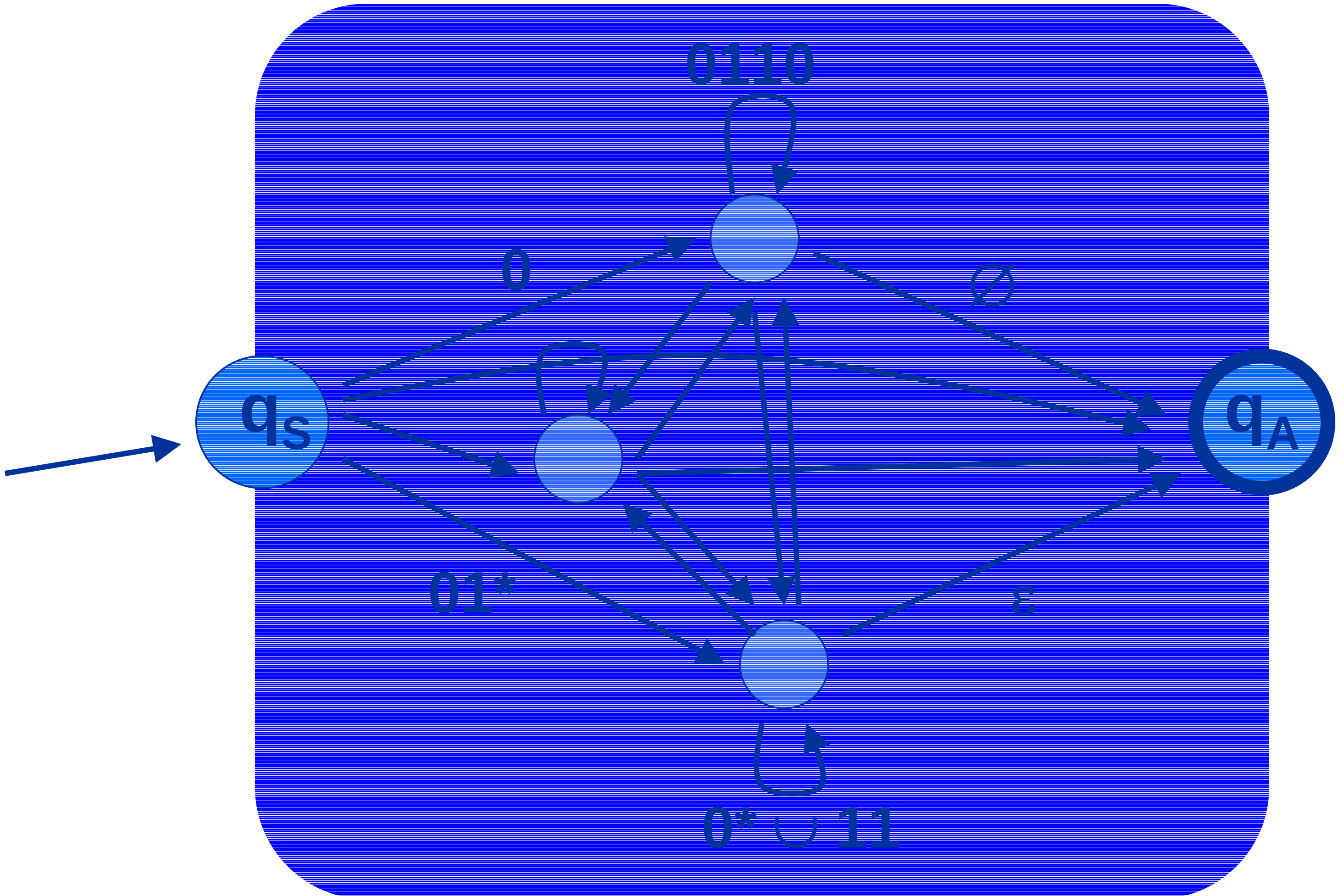
L = {w | w = $a^m b^n$, $m<10$, $n>10$}

Back to RL ~ RE

- The second part (Lemma 1.60):
  If a language is regular, then it can be described by a regular expression.
- Proof strategy:
  - regular implies equivalent DFA.
  - convert DFA to GNFA (generalized NFA)
  - convert GNFA to NFA.

GNFA: NFA that have regular expressions as transition labels

Example GNFA



Generalized NFA - defn

Generalized non-deterministic finite automaton $M=(Q, \Sigma, \delta, q_{start}, q_{accept})$ with

- Q finite set of states
- Σ the input alphabet
- q_{start} the start state
- q_{accept} the (unique) accept state
- $\delta:(Q - \{q_{accept}\})\times(Q - \{q_{start}\}) \rightarrow \mathcal{R}$ is the transition function

($\mathcal{R}$ is the set of regular expressions over Σ)

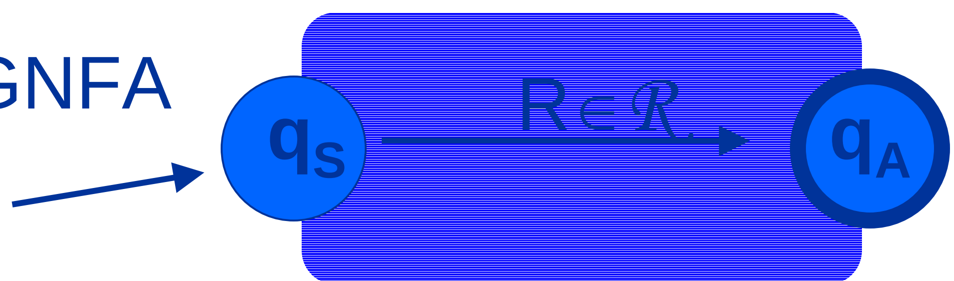
(NOTE THE NEW DEFN OF δ)

Characteristics of GNFA's δ

- $\delta:(Q \setminus \{q_{accept}\}) \times (Q \setminus \{q_{start}\}) \rightarrow \mathcal{R}$

The interior $Q \setminus \{q_{accept}, q_{start}\}$ is fully connected by δ
From q_{start} only 'outgoing transitions'
To q_{accept} only 'ingoing transitions'
Impossible $q_i \rightarrow q_j$ transitions are labeled "$\delta(q_i, q_j) = \emptyset$"

Observation: This GNFA recognizes the language L(R)



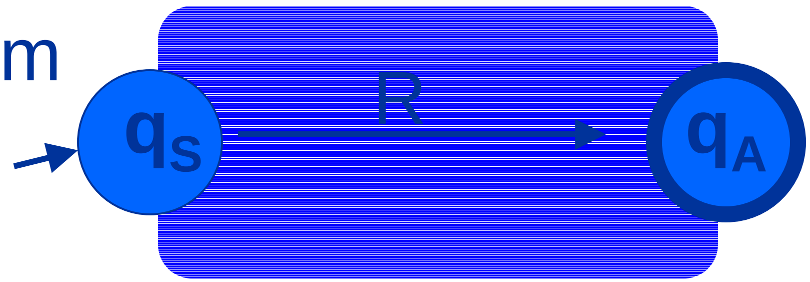
Proof Idea of Lemma 1.60

Proof idea (given a DFA M):

Construct an equivalent GNFA M' with k≥2 states

Reduce one-by-one the internal states until k=2

This GNFA will be of the form



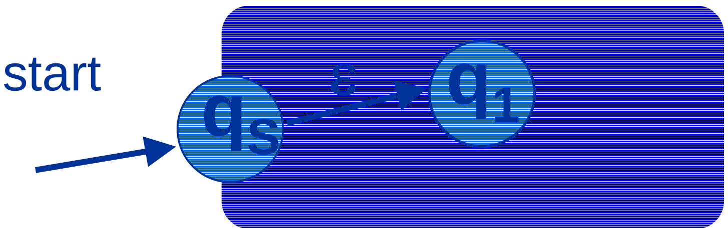
This regular expression R will be such that L(R) = L(M)

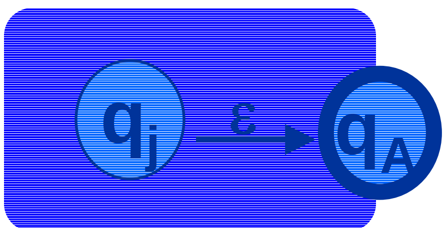
DFA M → Equivalent GNFA M’

Let M have k states $Q=\{q_1,\ldots,q_k\}$

- Add two states q_{accept} and q_{start}

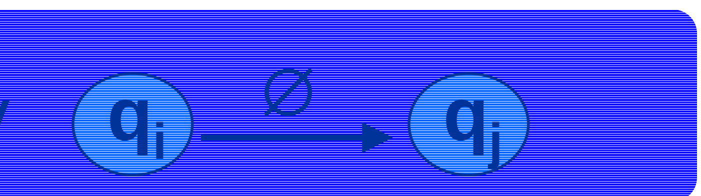
- Connect q_{start} to earlier q_1:



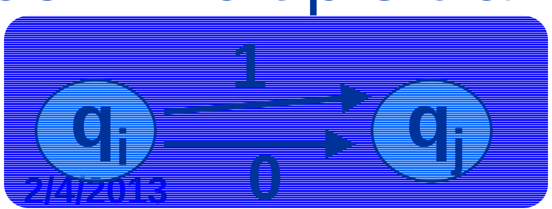


- Connect old accepting states to q_{accept}

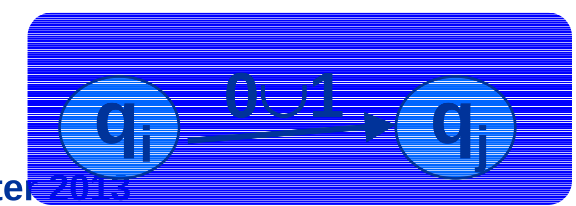
- Complete missing transitions by



- Join multiple transitions:



becomes



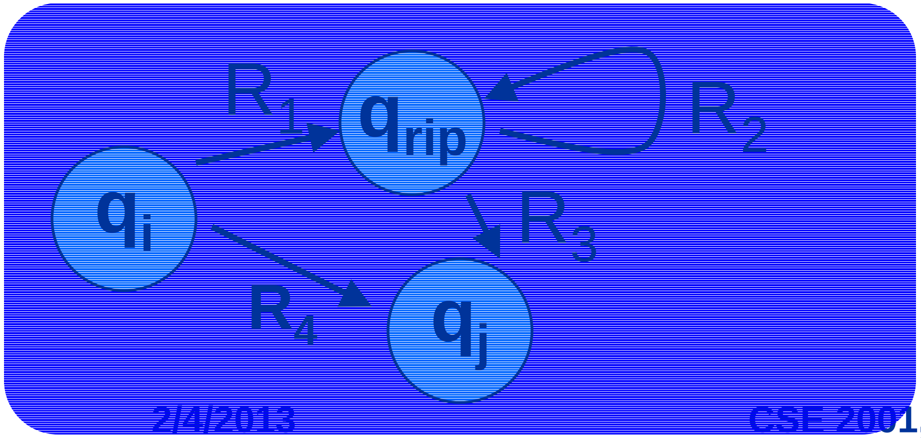
Remove Internal state of GNFA

If the GNFA M has more than 2 states, 'rip' internal q_{rip} to get equivalent GNFA M' by:

- Removing state q_{rip}: $Q' = Q \setminus \{q_{rip}\}$
- Changing the transition function δ by

$$\delta'(q_i,q_j) = \delta(q_i,q_j) \cup (\delta(q_i,q_{rip})(\delta(q_i,q_j))^*\delta(q_{rip},q_j))$$

for every $q_i \in Q' \setminus \{q_{accept}\}$ and $q_j \in Q' \setminus \{q_{start}\}$



=

Proof Lemma 1.60

Let M be DFA with k states

Create equivalent GNFA M' with k+2 states

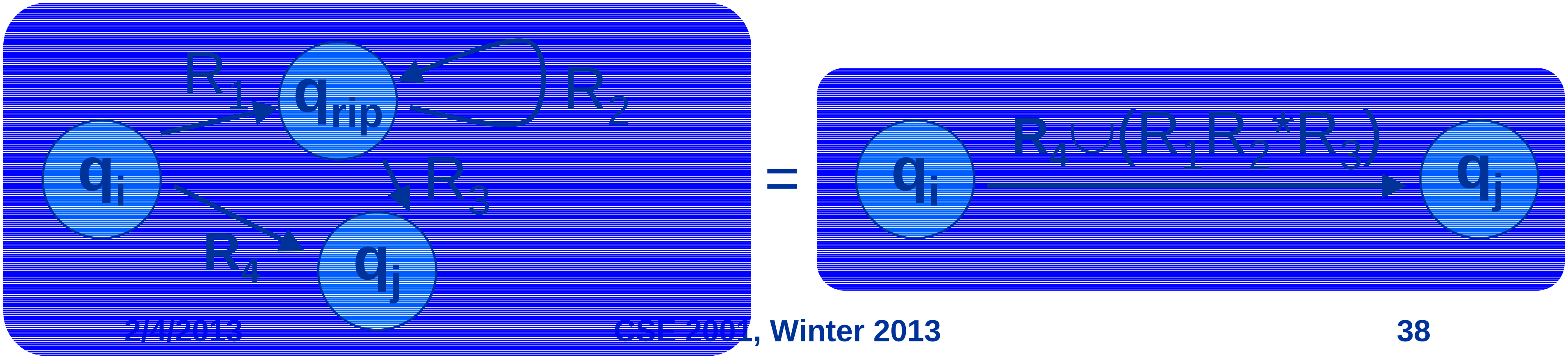
Reduce in k steps M' to M'' with 2 states

The resulting GNFA describes a single regular expressions R

The regular language L(M) equals the language L(R) of the regular expression R

Proof Lemma 1.60 - continued

- Use induction (on number of states of GNFA) to prove correctness of the conversion procedure.
- Base case: k=2.
- Inductive step: 2 cases – q_{rip} is/is not on accepting path.



Recap RL = RE

Let R be a regular expression, then there exists an NFA M such that L(R) = L(M)

The language L(M) of a DFA M is equivalent to a language L(M') of a GNFA = M', which can be converted to a two-state M''

The transition $q_{start} \xrightarrow{R} q_{accept}$ of M'' obeys L(R) = L(M'')

Hence: RE $\subseteq$ NFA = DFA $\subseteq$ GNFA $\subseteq$ RE