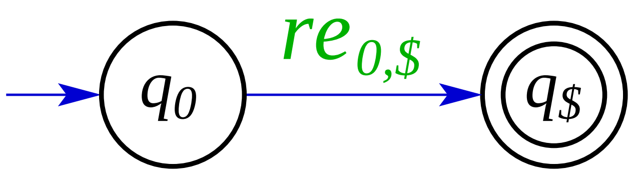
Regular Expressions and Non-Regular Languages

Mark Greenstreet, CpSc 421, Term 1, 2008/09

Lecture Outline

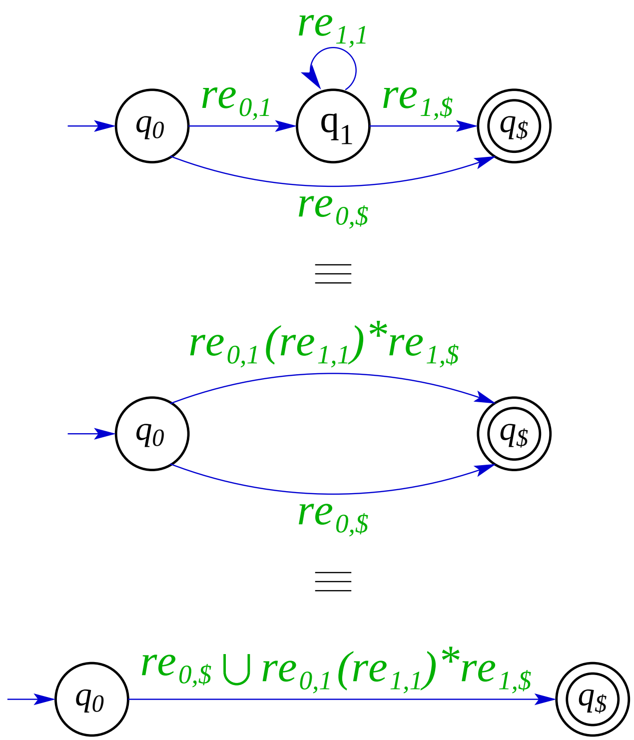
- Finishing the proof that the set of languages generated by regular expressions is the set of regular language.
  - In the Sept. 17 lecture, we showed that every language generated by a regular expression is a regular language.
    - Given a regular expression, R, we constructed an NFA, N, such that $L(N) = L(R)$. Because $L(N)$ is regular, so is $L(R)$.
  - Today, we will show that every regular language can be generated by a regular expression.
    - Given a regular language, A, we know that there is some DFA, M, that recognizes A. We will construct a regular expression, R, such that $L(R) = L(M)$.
- A language that is not regular.

From DFAs to REs



- Observation: DFA edges are labeled with symbols.
  A symbol or set symbols corresponds to a regular expression.
- Proof Idea: treat DFA edges as regular expressions.
  - If edge (q_i, q_j) is labeled with regular expression $re_{i,j}$, that means that the machine can move from state q_i to q_j by reading any string that matches $re_{i,j}$.
  - In general, such a machine isn't a DFA.
    Sispser calls this a GNFA, and we'll do the same.
  - If a GNFA has only two states, an initial state q_0 and a final state $q_\$$, where $q_\$$ is accepting and q_0 is not, then the language recognized by the GNFA is the language generated by the regular expression for edge $(q_0, q_\$)$.

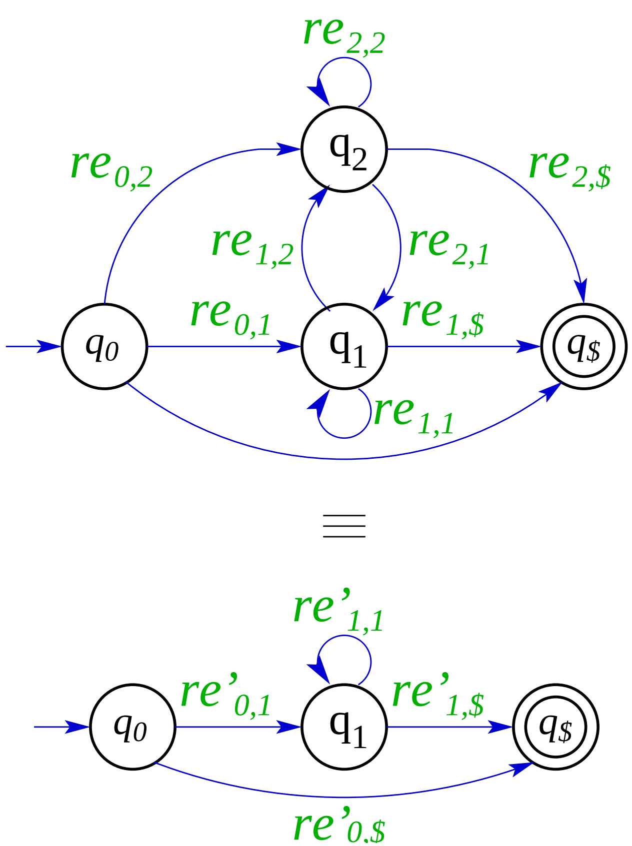
A GNFA with one intermediate state



We eliminate the state by accounting for all paths through the state.

In this case, the only such path is one the one from q_0 to $q_\$$.

A GNFA with two intermediate states



- $$re'_{0,1} = re_{0,1} \cup \left(re_{0,2} \cdot re^*_{2,2} \cdot re_{2,1}\right) \quad\Big|\quad re'_{0,\$} = re_{0,\$} \cup \left(re_{0,2} \cdot re^*_{2,2} \cdot re_{2,\$}\right)$$
  $$re'_{1,1} = re_{1,1} \cup \left(re_{1,2} \cdot re^*_{2,2} \cdot re_{2,1}\right) \quad\Big|\quad re'_{1,\$} = re_{1,\$} \cup \left(re_{1,2} \cdot re^*_{2,2} \cdot re_{2,\$}\right)$$

Defining GNFAs

Let $\mathcal{R}(\Sigma)$ denote the set of all regular expressions with alphabet Σ.

Let $(Q, \Sigma, \lambda, q_0, q_\$)$ be a GNFA where
$\lambda : (Q - \{\$\}) \times (Q - \{q_0\}) \rightarrow \mathcal{R}(\Sigma)$ is a labeling of transitions with regular expressions.

- Note: λ provides a label for every pair of states (that doesn't start with $q_\$$ or end with q_0).

  If there are no paths from q_i to q_j, then $\lambda(q_i, q_j) = \emptyset$.

Let G be a GNFA, the language recognized by G, $L(G)$ is the set of all strings s, such that

- There exists string $y_1, y_2, \ldots y_m$ such that $s = y_1 \cdot y_2 \cdots y_m$;
- There exists states $r_0, r_1, \ldots r_m$ such that:
  - $r_0 = q_0$;
  - y_i is generated by $\lambda(r_{i-1}, r_i)$.;
  - $r_m = q_\$$.

Shrinking a GNFA

Let $G_k = (Q_k, \Sigma, \lambda_k, q_0, q_\$)$ be a GNFA with $Q = \{q_0, q_1, \ldots q_k, q_\$\}$.

If $k > 0$, let $Q_{k-1} = Q - \{q_k\}$.

For $q_i, q_j \in Q_{k-1}$, let

$$\lambda_{k-1}(q_i, q_j) \quad = \quad \lambda_k(q_i, q_j) \cup (\lambda_k(q_i, q_k) \cdot \lambda_k(q_k, q_k)^* \cdot \lambda_k(q_k, q_j))$$

Let $G_{k-1} = (Q_{k-1}, \Sigma, \lambda_{k-1}, q_0, q_\$)$.

Claim: $L(G_{k-1}) = L(G_k)$.

$L(G_{k-1}) \subseteq L(G_k)$

Proof sketch:

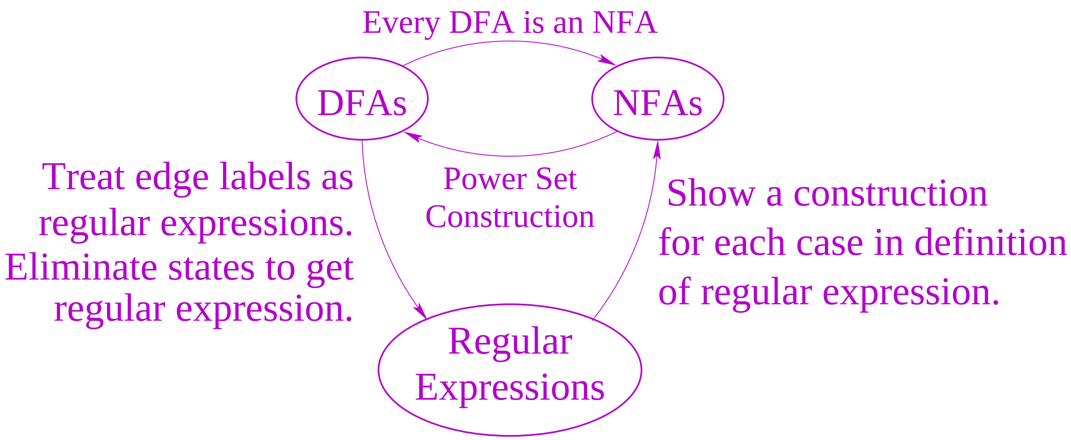
- For any $s \in L(G_{k-1})$, we can find $y_1 \ldots y_m$ be strings and $r_0 \ldots r_m$ that satisfy the acceptance conditions from slide 6.
- For each "transition" that G_{k-1} makes for these sequences:
  - If G_k can make the same "transition", we have G_k do that.
  - Otherwise, the transition must correspond to a regular expression for going from q_i to q_k and on to q_j. We construct a sequence of transitions for G_k that does the same thing.
- This gives us a seqence of strings $y'_1 \ldots y'_{m'}$ and a sequence of states $r'_0 \ldots r'_{m'}$ that show that G_k accepts s.
- For more details, see slides 13 through 16.

$L(G_{k-1}) \supseteq L(G_k)$

The proof is similar to the $L(G_{k-1} \subseteq L(G_k)$ case. Sketch:

- For any $s \in L(G_k)$, we can find $y_1 \ldots y_m$ be strings and $r_0 \ldots r_m$ that satisfy the acceptance conditions from slide 6.
- Now, the special case is when G_k makes a transition to state q_k (which doesn't exists for G_{k-1}.
  - We note that $q_k \neq r_0$, and $q_k \neq q_\$$.
  - Thus, we can find a sequence of transitions for G_k that starts in a state other than q_k, ends in a state other than q_k, where all of the states in the middle are q_k.
  - G_{k-1} can read the string for that entire sequence of transitions of G_k in a single move. This follows directly from how we accounted for moves through q_k when construcing the labels for G_{k-1}. for going from q_i to q_k and on to q_j. We construct a sequence of transitions for G_k that does the same thing.
- This gives us a seqence of strings $y'_1 \ldots y'_{m'}$ and a sequence of states $r'_0 \ldots r'_{m'}$ that show that G_{k-1} accepts s.
- I might add slides with details later.

RE = DFA = NFA



- Last Friday, we showed that every DFA is an NFA.
  - On Monday, we showed that every NFA is a DFA.
  - On Wednesday, we showd that every regular expression generates a language recognized by an NFA.
  - Today, we showed that every DFA recognizes a language that can be generated by a regular expression.

$\therefore$ DFAs, NFAs and regular expressions all describe the same set of languages.

$\square$

A non-regular langauge: $\mathbf{a}^n\mathbf{b}^n$

Discuss in class.

A non-regular langauge: $\mathbf{a}^n\mathbf{b}^n$

Proof by contradiction:

If $\mathsf{a}^n\mathsf{b}^n$ were are regular language, then there would be some DFA, M, that recognizes it. For the sake of contradiction, assume that such a machine exists.

M has some fixed number of states. Let k be this number.

Consider the string a^k. M visits $k + 1$ states from its initial state through reading a^k (including both the initial state and the state reached after reading a^k.

Therefore, there is at least one state that M visits at least twice (the “Pigeon Hole” principle).

Thus we can find i and j with $0 \leq i, j \leq k$ and $i \neq j$ such that M is in the same stae after reading a^i as it is after reading a^j.

This means that strings $\mathsf{a}^i\mathsf{b}^i$ and $\mathsf{a}^j\mathsf{b}^i$ bring M to the same state. Therefore, either M accepts both $\mathsf{a}^i\mathsf{b}^i$ and $\mathsf{a}^j\mathsf{b}^i$ or it rejects them both.

However, $\mathsf{a}^i\mathsf{b}^i$ is in the language and $\mathsf{a}^i\mathsf{b}^j$ is not.

Therefore, M cannot recognize the language $\mathsf{a}^n\mathsf{b}^n$.

The coming week

Reading:

September 19 (Today): Nonregular Languages – Read *Sipser* 1.4.
Lecture will cover through Example 1.73 (i.e. pages 77-80).

September 22 (Monday): Pumping Lemma Examples.
The rest of *Sipser* 1.4 (i.e. pages 80–82).

September 24 (Wednesday): Introduction to Context Free Languages – *Sipser* 2.1.
Lecture will cover through "Designing Context-Free Grammars" (i.e. pages 99-105).

September 26 (A week from today): Chomsky Normal Form
The rest of *Sipser* 2.1 (i.e. pages 105–109).

Homework:

September 19 (Today): Homework 1 due. Homework 2 goes out (on the web, later today, due Sept. 26).

September 26 (A week from Today): Homework 2 due. Homework 3 goes out (due Oct. 3).
The due date for homework 3 will be strict – no late assignments will be accepted.

Midterm: Oct. 8

$L(G_{k-1}) \subseteq L(G_k)$

Proof details:

- Let $s \in L(G_{k-1})$. Let $y_1 \ldots y_m$ be strings and $r_0 \ldots r_m$ be states that show that $s \in L(G_{k-1})$ as specified on slide 6.
- Our strategy now is to find a sequence of strings and states that show that $s \in L(G_k)$.
  - The intuitive idea is that a transition from q_i to q_j by G_{k-1} either corresponds to the same transition for G_k, or G_k goes from q_i to q_k, performs zero or more self-loops at q_k and then transitions to q_j.
  - Thus, each transition of G_{k-1} corresponds to either one or three steps of G_k.
  - We'll define $f(n)$ to map step numbers of G_{k-1} to step numbers of G_k.

$L(G_{k-1}) \subseteq L(G_k)$ (cont)

- $f(1) = 1$.
- For each $1 \leq i \leq m$
  - Note that $y_i \in L(\lambda_{k-1}(r_{i-1}, r_i))$, and that $\lambda_{k-1}(r_{i-1}, r_i) = \lambda_k(r_{i-1}, r_i) \cup (\lambda_k(r_{i-1}, q_k) \cdot \lambda_k(q_k, q_k)^* \cdot \lambda_k(q_k, r_i))$
  - If $y_i \in L(\lambda_k(r_{i-1}, r_i))$, let

$$\begin{aligned} y'_{f(i)} &= y_i \\ r'_{f(i)} &= r_i \\ f(i+1) &= f(i) + 1 \end{aligned}$$

  - Otherwise, $y_i \in L(\lambda_k(r_{i-1}, q_k) \cdot \lambda_k(q_k, q_k)^* \cdot \lambda_k(q_k, r_i))$, and (continued on next slide)

$L(G_{k-1}) \subseteq L(G_k)$ (cont)

- For each $1 \leq i \leq m$
  - If $y_i \in L(\lambda_k(r_{i-1}, q_k) \cdot \lambda_k(q_k, q_k)^* \cdot \lambda_k(q_k, r_i))$, then
    - There are strings z_0, z_1, ..., z_h such that

$$
\begin{array}{lcll}
y_i & = & z_0 \cdot z_1 \cdots z_h; & \\
z_0 & \in & L(\lambda_k(r_{i-1}, q_k)); & \\
z_d & \in & L(\lambda_k(q_k, q_k)), & \text{for all } 1 \leq d < h; \\
z_h & \in & L(\lambda_k(q_k, r_i)). &
\end{array}
$$

    - Let

$$
\begin{array}{lcll}
y'_{f(i)+d} & = & z_d, & \text{for all } 0 \leq d \leq h; \\
r'_{f(i)+d} & = & q_k, & \text{for all } 0 \leq d < h; \\
r'_{f(i)+h} & = & r_i; & \\
f(i+1) & = & f(i+j) &
\end{array}
$$

$L(G_{k-1}) \subseteq L(G_k)$ (cont)

- The sequences of strings $y'_1 \ldots y'_{f(m)}$ and states $r'_0 \ldots r'_{f(m)}$ satisfy the conditions for GNFA acceptance (slide 6).
- Thus, G_k accepts s.