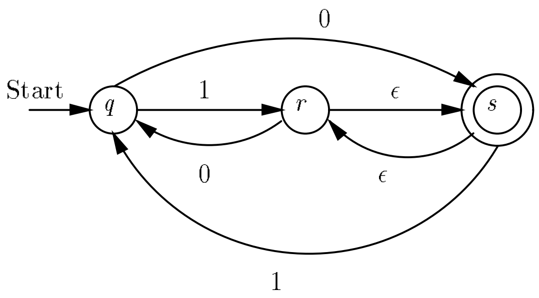
Finite Automata With ϵ-Transitions

Allow ϵ to be a label on arcs.

- Nothing else changes: acceptance of w is still the existence of a path from the start state to an accepting state with label w.
  - ❖ But ϵ can appear on arcs, and means the empty string (i.e., no visible contribution to w).

Example

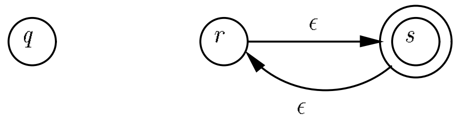


- 001 is accepted by the path q, s, r, q, r, s, with label $0\epsilon01\epsilon = 001$.

Elimination of ϵ-Transitions

ϵ-transitions are a convenience, but do not increase the power of FA's. To eliminate ϵ-transitions:

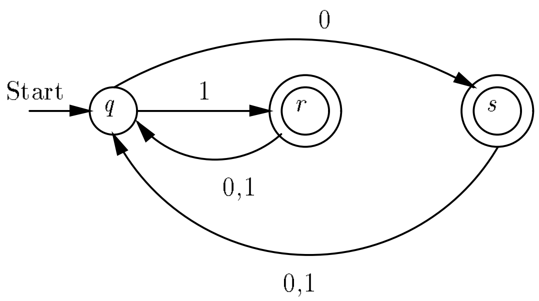
1. Compute the transitive closure of the ϵ arcs only.
   - ❖ Example:



$q \rightarrow \{q\}$; $r \rightarrow \{r, s\}$; $s \rightarrow \{r, s\}$.

2. If a state p can reach state q by ϵ-arcs, and there is a transition from q to r on input a (not ϵ), then add a transition from p to r on input a.
3. Make state p an accepting state if p can reach some accepting state q by ϵ-arcs.
4. Remove all ϵ-transitions.

Example



Regular Expressions

An algebraic equivalent to finite automata.

- Used in many places as a language for describing simple but useful patterns in text.

Operators and Operands

If E is a regular expression, then $L(E)$ denotes the language that E stands for. Expressions are built as follows:

- An operand can be:
  1. A variable, standing for a language.
  2. A symbol, standing for itself as a *set of strings*, i.e., **a** stands for the language $\{a\}$ (formally, $L(\mathbf{a}) = \{a\}$).
  3. ϵ, standing for $\{\epsilon\}$ (a language).
  4. $\emptyset$, standing for $\emptyset$ (the empty language).
- The operators are:
  1. $+$, standing for union. $L(E+F) = L(E) \cup L(F)$.
  2. Juxtaposition (i.e., no operator symbol, as in xy to mean $x \times y$) to stand for *concatenation*. $L(EF) = L(E)L(F)$, where the concatenation of languages L and M is $\{xy \mid x \text{ is in } L \text{ and } y \text{ is in } M\}$.
  3. $*$ to represent *closure*. $L(E^*) = \big(L(E)\big)^*$, where $L^* = \{\epsilon\} \cup L \cup LL \cup LLL \cup \cdots$.
- Parentheses may be used to alter grouping, which by default is $*$ (highest precedence), then concatenation, then union (lowest precedence).

Examples

- $L(\mathbf{001}) = \{001\}$.
- $L(\mathbf{0} + \mathbf{10}^*) = \{0, 1, 10, 100, 1000, \ldots\}$.
- $L\Big(\big(\mathbf{0}(\mathbf{0} + \mathbf{1})\big)^*\Big)$ = the set of strings of 0's and 1's, of even length, such that every odd position has a 0.

Equivalence of FA Languages and RE Languages

- We'll show an NFA with ϵ-transitions can accept the language for a RE.
- Then, we show a RE can describe the language of a DFA (same construction works for an NFA).
- The languages accepted by DFA, NFA, ϵ-NFA, RE are called the *regular* languages.

RE to ϵ-NFA

- Key idea: construction of an ϵ-NFA with one accepting state is by induction on the height of the expression tree for the RE.
- Pictures of the basis and inductive constructions are in the course reader.

Example

We'll go over the general construction in class and work the example of $\big(\mathbf{0}(\mathbf{0} + \mathbf{1})\big)^*$.

FA-to-RE Construction

Two algorithms:

1. *State elimination*: gives smaller expression, in general, and easier to apply. Covered in course reader.
2. A simple, inductive construction, which we'll do here (also in reader).

- Let A be a FA with states $1, 2, \ldots, n$.
- Let $R_{ij}^{(k)}$ be a RE whose language is the set of labels of paths that go from state i to state j without *passing through* any state numbered above k.
- Construction, and the proof that the expressions for these RE's are correct, are inductions on k.

Basis: $k = 0$. Path can't go through *any* states.

- Thus, path is either an arc or the null path (a single node).
- If $i \neq j$, then $R_{ij}^{(0)}$ is the sum of all symbols a such that A has a transition from i to j on symbol a ($\emptyset$ if none).
- If $i = j$, then add ϵ to above.

Induction: Assume we have correctly developed expressions for the $R^{(k-1)}$'s. Then for the $R^{(k)}$'s:

- $R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)}(R_{kk}^{(k-1)})^* R_{kj}^{(k-1)}$

Proof it works: A path from i to j that goes through no state higher than k either:

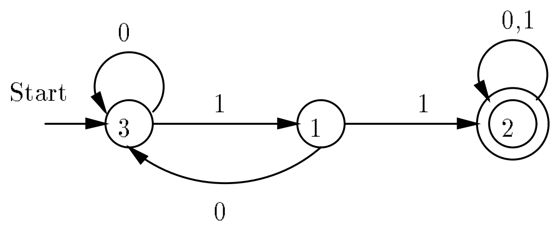
1. Never goes through k, in which case the path's label is (by the IH) in the language of $R_{ij}^{(k-1)}$, or
2. Goes through k one or more times. In this case:
   - $R_{ik}^{(k-1)}$ contains the portion of the path that goes from i to k for the first time.
   - $(R_{kk}^{(k-1)})^*$ contains the portion of the path (possibly empty) from the first k visit to the last.
   - $R_{kj}^{(k-1)}$ contains the portion of the path from the last k visit to j.

Final step: The RE for the entire FA is the sum (union) of the RE's $R_{ij}^{(n)}$, where i is the start state and j is one of the accepting states.

- Note that superscript (n) represents no restriction on the path at all, since n is the highest-numbered state.

Example

The following is the "clamping" automaton, with states named by integers:



Some basis expressions:

- $R_{11}^{(0)} = \epsilon$.

- $R_{12}^{(0)} = \mathbf{1}$.
- $R_{22}^{(0)} = \epsilon + \mathbf{0} + \mathbf{1}$.
- $R_{31}^{(0)} = \mathbf{1}$.
- $R_{32}^{(0)} = R_{21}^{(0)} = \emptyset$.

Two inductive examples:

- $R_{32}^{(1)} = R_{32}^{(0)} + R_{31}^{(0)}(R_{11}^{(0)})^* R_{12}^{(0)} = \emptyset + \mathbf{1}\epsilon^*\mathbf{1} = \mathbf{11}$.
  - Uses *algebraic laws*: $\epsilon^* = \epsilon$; $R\epsilon = \epsilon R = R$ (ϵ is the identity for contatenation); $\emptyset + R = R + \emptyset = R$ ($\emptyset$ is the identity for union).
- $R_{22}^{(1)} = R_{22}^{(0)} + R_{21}^{(0)}(R_{11}^{(0)})^* R_{12}^{(0)} = \epsilon + \mathbf{0} + \mathbf{1} + \emptyset\epsilon^*\mathbf{1} = \epsilon + \mathbf{0} + \mathbf{1}$.
  - Additional algebraic law used: $\emptyset R = R\emptyset = \emptyset$ ($\emptyset$ is the annihilator for concatenation).