Finite Automata With $\epsilon$-Transitions

Allow $\epsilon$ 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 $\epsilon$ can appear on arcs, and means the empty string (i.e., no visible contribution to $w$).

Example

![Diagram of a finite automaton with $\epsilon$-transitions]

- 001 is accepted by the path $q, r, q, r, s$, with label $0\epsilon0\epsilon = 001$.

Elimination of $\epsilon$-Transitions

$\epsilon$-transitions are a convenience, but do not increase the power of FA’s. To eliminate $\epsilon$-transitions:

1. Compute the transitive closure of the $\epsilon$ arcs only.
   - Example:

   ![Diagram of transitive closure]

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

2. If a state $p$ can reach state $q$ by $\epsilon$-arcs, and there is a transition from $q$ to $r$ on input $a$ (not $\epsilon$), 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 $\epsilon$-arcs.

4. Remove all $\epsilon$-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, $L(a) = \{a\}$).
  3. $\epsilon$, 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 | x is in $L$ and y is in $M$\}.
  3. $*$ to represent closure. $L(E^*) = (L(E))^*$,
     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).

2
Examples

- \( L(001) = \{001\} \).
- \( L(0 + 10^*) = \{0, 1, 10, 100, 1000, \ldots\} \).
- \( L\left((0(0 + 1))^*\right) = \) 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 \( \epsilon \)-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, \( \epsilon \)-NFA, RE are called the regular languages.

RE to \( \epsilon \)-NFA

- Key idea: construction of an \( \epsilon \)-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 \( (0(0 + 1))^* \).

FA-to-RE Construction

Two algorithms:

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 \( R_{ij}^{(k)} \) 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 \( \epsilon \) 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.
   - \( (R_{ik}^{(k-1)})^* \) 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:

![Diagram of the automaton](image)

Some basis expressions:

• \( R_{11}^{(n)} = \epsilon \).
\begin{itemize}
    \item $R^{(0)}_{12} = 1$.
    \item $R^{(0)}_{22} = \varepsilon + 0 + 1$.
    \item $R^{(0)}_{31} = 1$.
    \item $R^{(0)}_{32} = R^{(0)}_{21} = \emptyset$.
\end{itemize}

Two inductive examples:
\begin{itemize}
    \item $R^{(1)}_{32} = R^{(0)}_{32} + R^{(0)}_{31} (R^{(0)}_{11})^* R^{(0)}_{12} = \emptyset + 1\varepsilon 1 = 11$.
        \begin{itemize}
            \item Uses algebraic laws: $\varepsilon^* = \varepsilon$; $R\varepsilon = \varepsilon R = R$ ($\varepsilon$ is the identity for concatenation);
            $\emptyset + R = R + \emptyset = R$ ($\emptyset$ is the identity for union).
        \end{itemize}
    \item $R^{(1)}_{22} = R^{(0)}_{22} + R^{(0)}_{21} (R^{(0)}_{11})^* R^{(0)}_{12} = \varepsilon + 0 + 1 + \emptyset\varepsilon 1 = \varepsilon + 0 + 1$.
        \begin{itemize}
            \item Additional algebraic law used: $\emptyset R = R\emptyset = \emptyset$ ($\emptyset$ is the annihilator for concatenation).
        \end{itemize}
\end{itemize}