Regular Language Algorithms
We have seen many algorithms for manipulating computational systems for the regular languages. Each treats these computational systems as data that can be manipulated. It is worthwhile trying to catalog these algorithms.
Trivial algorithms

- The following algorithms are trivial
  - DFA to NFA
  - NFA to \(\Lambda\)-NFA
  - NFA to GenNFA

A DFA is a quintuple \(D=(Q, \Sigma, s, F, \delta)\)
\[
\delta: Q \times \Sigma \longrightarrow Q
\]

An NFA is a quintuple \(N=(Q, \Sigma, s, F, T)\)
\[
T: Q \times \Sigma \longrightarrow \text{Set}(Q)
\]

A \(\Lambda\)-NFA is a quintuple \(L=(Q, \Sigma, s, F, \delta)\)
\[
\delta: Q \times (\Sigma \cup \Lambda) \longrightarrow \text{Set}(Q)
\]

GenNFA is a quintuple \(A=(Q, \Sigma, s, F, \delta)\)
\[
\delta: Q \times Q \longrightarrow \text{RegExp}(\Sigma)
\]

\(Q\) is a set of states
\(S\) is a set of characters (the alphabet)
\(S\) is a State called the start state
\(F\) is a set of states called the final states
Closure Algorithms over DFAs

- **Reversal**
  - Reverse arrows
  - Start becomes final, final becomes start

- **Complement**
  - Swap the status of final and non-final states

- **Union**
  - Product construction
  - Any pair with at least one final state is final

- **Intersection**
  - Product construction
  - Only the pair with both final states is final

- **Kleene Star**
  - New start and final states, \( \Lambda \)-transitions, and loop backs.
  - DFA to \( \Lambda \)-NFA
Transformations

- **NFA to DFA**
  - Subset construction, union of delta ranges
- **DFA to Minimal DFA**
  - Partition states into equivalence classes
- **Reg Exp to NFA**
  - Via GenNFA where transitions are on RE
  - Start with 2 states with RE as transition
  - Add new states by structure of RE on transition
- **DFA to Reg Exp (Via GenNFA)**
  - By state removal,
  - Remove states one at a time, and add RE to transitions
Acceptance algorithms

• DFA
  – Delta extension
  – Last state is amongst final states

• NFA
  – Delta extension to sets of states

• Reg Exp
  – Via empty predicate and derivatives of RE