Formal Definition of a Generalized Nondeterministic Finite Automaton (GNFA)

Given an alphabet Σ, let *RegExp* denote the set of all regular expressions over Σ.

A *generalized nondeterministic finite automaton* G is a 5-tuple $(Q, \Sigma, \delta, q_{\text{start}}, q_{\text{accept}})$, where

- Q is a finite set of states;
- Σ is a finite alphabet;
- $\delta : (Q - \{q_{\text{accept}}\}) \times (Q - \{q_{\text{start}}\}) \longrightarrow \mathit{RegExp}$ is the transition function;
- $q_{\text{start}} \in Q$ is the start state; and
- $q_{\text{accept}} \in Q$ is the accept state.

A GNFA is like an NFA except the arrows are labeled with arbitrary regular expressions, not just ε or alphabet symbols.

Furthermore, the above definition imposes the following additional conditions:

- The start state has arrows going to every other state but has no incoming arrows.
- There is exactly one accept state and it has arrows coming from every other state but no outgoing arrows.
- Every non-start, non-accept state has arrows going to and coming from every other such state.

Computation Performed by a GNFA

A GNFA $G = (Q, \Sigma, \delta, q_{\text{start}}, q_{\text{accept}})$ *accepts* a string $w \in \Sigma^*$ if $w = w_1 w_2 \ldots w_k$, where each $w_i \in \Sigma^*$ and a corresponding sequence of states $q_0, q_1, q_2, \ldots, q_k \in Q$ exists such that

1. $q_0 = q_{\text{start}}$;
2. $q_k = q_{\text{accept}}$; and
3. for each $i = 1, 2, \ldots, k$, $w_i \in L(R_i)$, where $R_i = \delta(q_{i-1}, q_i)$.

This last condition means that R_i is the regular expression on the arrow from state q_{i-1} to state q_i.

The *language recognized by* G is $L(G) = \{w \mid G \text{ accepts } w\}$.