- Gramática: permite gerar todas as palavras da linguagem que represento.

$$G = (V, T, P, S)$$

↳ V: conjunto finito de variáveis (ou símbolos não terminais), usados para gerar palavras.
↳ T: conjunto finito de símbolos terminais, disjuntos de V. Símbolos que aparecem na palavra final.
↳ P: conjunto de produções, ou regra de derivação, define como as palavras podem ser substituídas
↳ S: símbolo inicial, $S \in V$.

Autômato Finito Determinístico (AFD)

- 5-upla: $M = (\Sigma, Q, \delta, q_0, F)$
  ↳ Σ: alfabeto
  ↳ Q: conjunto de todos os estados
  ↳ δ: função programa → $\delta: Q \times \Sigma \to Q$
  ↳ q_0: estado inicial → $q_0 \in Q$
  ↳ F: conjunto de estados finais → $F \subseteq Q$
- Função Programa estendida
  ↳ $\underline{\delta}: Q \times \Sigma^* \to Q$: define o estado após ler toda a palavra w tal que $w \in \Sigma^*$
    ↳ Ex: $\underline{\delta}(q, w)$ = estado resultante após ler toda palavra.
    ↳ Propriedades:
      * $\underline{\delta}(q, \varepsilon) = q$
      * $\underline{\delta}(q, aw) = \underline{\delta}(\delta(q, a), w)$ tal que $a \in \Sigma$ e $w \in \Sigma^*$

Autômato Finito Não-determinístico (AFN)

- Qualquer AFN pode ser simulado por um AFD.
- $\delta: Q \times \Sigma \to 2^Q$: ou seja δ recebe um par (estado, símbolo) e retorna um subconjunto de Q.
  ↳ Se em alguma linha da função programa tivermos um subconjunto de Q, esse autômato é um AFN.
- Função Programa Estendida: $\underline{\delta}: 2^Q \times \Sigma^* \to 2^Q$
  ↳ Seja $M = (\Sigma, Q, \delta, q_0, F)$ um AFN e $P \in 2^Q$:
    * $\underline{\delta}(P, \varepsilon) = P$
    * $\underline{\delta}(P, aw) = \underline{\delta}(\bigcup_{q \in P} \delta(q, a), w)$ (união de todos os conjuntos de estados de q que podem ser acessados a partir de cada estado de P lendo a)

Algoritmo AFN para AFD

- Entrada: AFN $(Q, \Sigma, \delta, q_0, F)$
  Saída: AFD $(Q', \Sigma, \delta', q_0', F')$
- 1°: o estado inicial do AFD é $\{q_0\}$
- 2°: enquanto houver estados não processados:
  ↳ para cada símbolo $a \in \Sigma$, calcule:
    * a união de $\delta(q, a)$ para todo q no conjunto atual
    * se esse novo conjunto não estiver na lista adicione.
- 3°: marque como finais os conjuntos que contêm algum estado final do AFN.

Autômato Finito com Movimentos Vazio (AFNε ou AFε)

- Transições sem leitura de símbolo da fita.
- Qualquer AFN com movimentos vazio pode ser simulado por um AFN
- Fecho-ε ou F_ε: seja $M = (\Sigma, Q, \delta, q_0, F)$ um AF_ε
  ↳ $F_\varepsilon: Q \to 2^Q$
    ↳ Se $\delta(q, \varepsilon)$ não é definido então $F_\varepsilon(q) = \{q\}$
    ↳ Se $\delta(q, \varepsilon)$ é definido $F_\varepsilon(q) = \{q\} \cup \delta(q, \varepsilon) \cup_{p \in \underline{\delta}(q, \varepsilon)} F_\varepsilon(p)$
      ↳ Conjunto de todos os estados que podem ser acessados com zero ou mais transições vazias (ε).
- Para conjunto de estados
  ↳ $F_\varepsilon: 2^Q \to 2^Q$ tq $F_\varepsilon(P) = \bigcup_{q \in P} F_\varepsilon(q)$, e $P \in 2^Q$
    ↳ União de $\{F_\varepsilon(q) \mid q \in P\}$

- Função Programa Estendido: $\underline{\delta}: 2^Q \times \Sigma^* \to 2^Q$
  ↳ $\underline{\delta}(P, \varepsilon) = F_\varepsilon(P)$ tq $P \in 2^Q$
  ↳ $\underline{\delta}(P, wa) = F_\varepsilon(R)^*$ (após achar R faz $F_\varepsilon(R)^*$)
    ↳ $R = \{r \mid r \in \delta(s, a)$ e $s \in \underline{\delta}(P, w)\}$
      ↳ onde: $a \in \Sigma$, $w \in \Sigma^*$

- Algoritmo $AF_\varepsilon \to$ AFN
  ↳ 1° → Calcular $F_\varepsilon(q)$ $\forall q \in Q$, todos os estados alcançáveis a partir de q usando apenas transições ε.
  ↳ 2° → para cada $q \in Q$ e cada $a \in \Sigma$:
    ↳ (I) Inicialize $R = \emptyset$
    ↳ (II) Para cada estado $p \in F_\varepsilon(q)$ adicione $\delta(p, a)$ em R (estados alcançados com a)
    ↳ (III) Faça:
      * $\delta'(q, a) = \bigcup F_\varepsilon(r)$ $\forall r \in R$, ou seja F_ε de todos os estados alcançado com a.
  ↳ 3° → Novos estados finais: um estado $q \in Q$ será final em F' se $(F_\varepsilon(q) \cap F) \neq \emptyset$, ou seja se o $F_\varepsilon(q)$ contém algum estado final original, então q também é final.

Expressões Regulares: Formalismo denotacional

- ∅ é ER, e denota a linguagem vazia
- ε é ER, e denota a linguagem $\{\varepsilon\}$
- x é ER, onde $x \in \Sigma$ e denota a linguagem $\{x\}$
- Se r e s são ER, e denotam as linguagens R e S respectivamente, então:
  ↳ (r+s) é ER, e denota R ∪ S
  ↳ (rs) é ER, e denota RS
  ↳ (r^*) é ER, e denota R^*
- Precedência de parênteses em ER:
  ↳ 1°: Concatenação sucessivo (* ou +)
  ↳ 2°: Concatenação
  ↳ 3°: União