Finite State Machines

- G is a finite SM if |X| < ∞.

Notations: E-NFSM, NFSM, DFSM.

- Language model equivalence of E-NFSM and NFSM:

Given E-NFSM G, construct NFSM G′ := (X, X, δ, X0, Xm)

α' (x, σ) := E* (α (E' (x), σ)) = α' (x, σ)

Xm := \{x_m \cup x_0 \text{ if } x_m \cap E' (x) \neq \emptyset \}

Then Lm (G′) = Lm (G) \land L (G′) = L (G).

Example:

![Diagram of G and G']

- Language model equivalence of NFSM and DFSM:

Given NFSM G := (X, X, δ, X0, Xm), construct DFSM

G := (X, X, δ', X0, Xm)

X := 2^X, Xm := {x \in X | x \cap Xm \neq \emptyset}

δ' (x, σ) := \bigcup_{x \in X} δ (x, σ)

Then Lm (G′) = Lm (G) \land L (G′) = L (G). Known as power set construction.

Exercise:

![Diagram of exercise]

Remark: A deterministic DES with finite states can always be modeled as a DFSM.
Regular Languages

- We will now characterize the class of languages that can be represented by FSMS. (useful for development of algorithms).

- **Regular Class** ($\mathcal{L}_R$): $\emptyset, \{\varepsilon\}, \{\varepsilon, \cdot \} \in \mathcal{L}_R \leq 2^\Sigma^*$

  $\mathcal{L}_R$:

  $K, K_1, K_2 \in \mathcal{L}_R \Rightarrow K_1 \cup K_2, K_1 \cdot K_2, K^* \in \mathcal{L}_R$

For simplicity of notation, regular expressions are used for regular langs.

- $\phi, \varepsilon, \cdot$ are regular expressions
- $r_1, r_2$ regular expressions $\Rightarrow r_1 + r_2, r_1 \cdot r_2, r_1^*$ regular expressions.

**Example:** $(a+b)^*$, $a^*b$, etc. are regular. $+a$, $a+b$ not regular.

**Proof:** ($\Rightarrow$) Label states of $G$ by $1, \ldots, m$. Define regular expressions inductively for each $i, j \leq m$:

$$ r_{ij}^0 = \begin{cases} 
\varepsilon & \text{if } i = j \\
\{ \sigma \in \Sigma | \alpha(i, \sigma) = j \} & \text{if } i \neq j 
\end{cases} $$

$$ r_{ij}^k = r_{ik}^{k-1} (r_{kk})^* r_{kj}^{k-1} + r_{ij}^{k-1} \quad \text{for } k \leq m $$

Then $L(r_{ij}^k)$ = set of strings starting from $i$, ending at $j$, and visiting states with label no larger than $k$.

Clearly, $L(r_{ij}^k)$ is regular and $L_m(G) = L(\bigcup_{j \in \{1, \ldots, m\}} r_{ij}^m)$.
Regular language: Equivalence with DFSM (ctnd.)

\( \iff \) Since DFSM is "knap. equivalent" to \( \epsilon \)-NFSM, it suffices to show existence of an \( \epsilon \)-NFSM \( G \) with \( L_\epsilon(G) = L(r) \). Shown by induction on number of operations in \( r \).

**Base step:** \( \# \text{ of operations} = 0 \)
\[ r = \emptyset, \text{ or } r = \epsilon, \text{ or } r = \tau_0 \text{ for some } \tau_0 \in \Sigma. \]

**Induction step:** Suppose \( r = r_1 + r_2 \) then

**Next suppose** \( r = r_1 \cdot r_2 \) then

**Finally suppose** \( r = r_1^* \) then

*Corollary:* Given a language model \( (K_m, K) \), there exists a DFSM \( G \) such that \( (L(G), L(G)) = (K_m, K) \) iff \( K_m \) and \( K \) both are regular.
Example:

\[ \begin{align*}
L_m(G) &= L(r_{11}^2) \\
\gamma_1^0 &= \epsilon \\
\gamma_1^2 &= a \\
\gamma_2^0 &= d \\
\gamma_2^0 &= \epsilon \\
\gamma_1^1 &= \gamma_1^0 \cdot (\gamma_1^0)^* \gamma_1^0 + \gamma_1^0 = \epsilon (\epsilon)^* \epsilon + \epsilon = \epsilon \\
\gamma_2^1 &= \gamma_2^0 \cdot (\gamma_2^0)^* \gamma_2^0 + \gamma_2^0 = \epsilon (\epsilon)^* a + a = a \\
\gamma_2^1 &= \gamma_2^1 \cdot (\gamma_2^1)^* \gamma_2^1 + \gamma_2^1 = d (\epsilon)^* d + d = d \\
\gamma_2^1 &= \gamma_2^1 \cdot (\gamma_2^1)^* \gamma_2^1 + \gamma_2^1 = d (\epsilon)^* a + \epsilon = da + \epsilon \\
\gamma_2^1 &= \gamma_2^1 \cdot (\gamma_2^1)^* \gamma_2^1 + \gamma_2^1 = a (da + \epsilon)^* d + \epsilon = a (da)^* d + \epsilon = (ad)^* \\
\end{align*} \]

Definitions:

\[ Re_G(x) = \text{set of reachable states from } x \text{ in } G \]

\[ = \{ x' \in X | \exists \sigma \in \Sigma^* \text{ s.t. } x' \in a^*(x, \sigma) \} \]

• G is called accessible if \( Re_G(x_0) = X \)

• G is called co-accessible if \( \forall x \in X : Re_G(x) \land X_{m} \neq \phi \)

• G is trim if it is accessible + co-accessible.

• It is always possible to construct a language equivalent trim state machine for any given state machine.