HW #2: Re-write

Perfect Shuffle: \( P(A,B) = \{a_1b_1, a_2b_2, \ldots, a_kb_k \mid a_1, a_2, \ldots, a_k \in A, b_1, b_2, \ldots, b_k \in B \}
\)
\( a_i, b_i \in \Sigma_i^B \)

Prove the perfect shuffle is closed under \( P(A,B) \)

**Proof:** Assume \( A, B \) are regular \( \Rightarrow \exists \) NFAs

\( N_A = (Q_A, \Sigma_A, q_0A, \delta_A, F_A) \) which recognizes \( A \)

\( N_B = (Q_B, \Sigma_B, q_0B, \delta_B, F_B) \) which recognizes \( B \)

Idea: we are going to create a NFA \( N_{PS} \) that combines \( N_A \) and \( N_B \). We would like to alternate between doing computation in \( N_A \) and \( N_B \) sequentially.

Here is a picture to describe the idea: we will expand the state space to keep track of the state of \( N_A \), \( N_B \) are currently in and which state they should be in next.

\( N_A \xrightarrow{\gamma A} \square \xrightarrow{\gamma A} \bigcirc \)

\( N_B \xrightarrow{\gamma B} \square \xrightarrow{\gamma B} \bigcirc \)

\( N_{PS} \)

\( \left( q_{0A}, q_{0B}, A \right) \xrightarrow{\delta_{PS}} \left( q_{0A}, q_{0B}, B \right) \)

\( \left( Q_A \times Q_B \times \Sigma_B^A \right) \quad A \text{ and } B \text{ keep track of computation} \)

\( \Sigma_{PS} = \Sigma_A^B \cup \Sigma_B^A \)

\( q_{ops} = (q_{0A}, q_{0B}, A) \)

\( F_{PS} = F_A \times F_B \times \Sigma_A^B \)

**Construction:**

We now construct the \( N_{PS} \) shown above.

\( N_{PS} = (Q_{PS}, \Sigma_{PS}, q_{ops}, \delta_{PS}, F_{PS}) \)

**Notes:**

- Assumption: \( A, B \) are regular NFAs.
- Diagram for \( N_{PS} \).
\[ \delta \text{ps} ((r, s, y), \sigma) = \begin{cases} \delta a (y, \sigma a), s, B \, \delta a \, \delta B (s, \sigma), A \, \delta a \, \delta B & \text{if } y = A \text{ and } \sigma \in \Sigma_B \\ \emptyset & \text{if } y = A \text{ and } \sigma \not\in \Sigma_B \\ \emptyset & \text{if } y = B \text{ and } \sigma \not\in \Sigma_B \\ \emptyset & \text{if } y = B \text{ and } \sigma \in \Sigma_B \\ \end{cases} \]

\[ \text{correctness of construction} \]

we must show \( \text{w is accepted by Nps} \)

\( \rightarrow \) if \( \text{w is accepted by Nps} \) then \( \text{w is accepted by a perfect shuffle} \). Since \( \text{w is accepted by Nps} \) then \( \text{w has the form ab, ab, ... ab} \), where \( a, b \in \Sigma_A \) and \( a, b \in \Sigma_B \). Nps will go back and forth computing in \( na \text{ and } nb \) with every other character. So we will start in state \((qa, qb, a)\) and the first part of this state will only transition on characters \( a, a, ... a \), similarly the \( \text{NFA NB} \) will compute on \( b, b, ... b \) since \( a, b \) at \( \sigma \). The computation for the first part will end in some state \( qa \), and likewise the computation for \( \text{NB} \) will end in \( qb \). We are ending computation in some state \((fa, fb, a)\) and the \text{a} will be in the \( \text{NFA} \) because we stopped on \( \text{a} \), so the next jump will be \( A \). This is an accepting state so therefore we accept.

\( \therefore \text{w is accepted by Nps} \) then \( \text{w is accepted by a perfect shuffle} \). If \( \text{w is accepted by Nps} \) then it must go along a computational path that alternates between computing in \( na \text{ and } nb \). Let's call the odd characters of \( w \) \( a \) and the even \( b \). So \( |w| \) is even so the first part of our states computes on the \( a \), started in \( qa \) and ended in \( fa \), similarly the second part of our states started in \( qb \) and ended in \( fb \), and \( b, b, ... b \) is accepted by \( \text{NPS} \) so \( \text{w is accepted by Nps} \). We have just shown that \( \text{w is accepted by Nps} \) iff \( \text{w is accepted by a perfect shuffle} \).