1 Closure Properties

Lemma 1 Let $A$ and $B$ be languages recognized by PDAs $M_A$ and $M_B$, respectively, then $A \cup B$ is also recognized by a PDA called $M_{A \cup B}$.

Proof: The graph of $M_{A \cup B}$ consists of the graphs of $M_A$ and $M_B$ plus a new start vertex $start_{A \cup B}$, which is joined by $\lambda$-edges to the start vertices $start_A$ and $start_B$ of $M_A$ and $M_B$, respectively. Its final vertices are the final vertices of $M_A$ and $M_B$. The graph is shown in figure 1.

![Figure 1: PDA $M_{A \cup B}$](image)

While it is clear that $L(M_{A \cup B}) = L(M_A) \cup L(M_B)$, we present the argument for completeness.
First, we show that $L(M_{A\cup B}) \subseteq L(M_A) \cup L(M_B)$. Let $w \in L(M_{A\cup B})$. Then there is a $w$-recognizing computation path from $start_{A\cup B}$ to a final vertex $f$. If $f$ lies in $M_A$, then removing the first edge of $P$ leaves a path $P'$ from $start_A$ to $f$. Further, at the start of $P'$, the stack is empty and nothing has been read, so $P'$ is a $w$-recognizing path in $M_A$. That is, $w \in L(M_A)$. Similarly, if $f$ lies in $M_B$, then $w \in L(M_B)$. Either way, $w \in L(M_{A\cup B})$.

Second, we show that $L(M_A) \cup L(M_B) \subseteq L(M_{A\cup B})$. Suppose that $w \in L(M_A)$. Then there is a $w$-recognizing computation path $P'$ from $start_A$ to a final vertex $f$ in $M_A$. Adding the $\lambda$-edge $(start_{A\cup B}, start_A)$ to the beginning of $P'$ creates a $w$-recognizing computation path in $M_{A\cup B}$, showing that $L(M_A) \subseteq L(M_{A\cup B})$. Similarly, if $w \in L(M_B)$, then $L(M_B) \subseteq L(M_{A\cup B})$.

Our next construction is simplified by the following technical lemma.

Lemma 2 Let PDA $M$ recognize $L$. There is a PDA $M'$ which recognizes $L$; further $M'$ has only one final vertex, final$_{M'}$, and $M'$ will always have an empty stack when it reaches final$_{M'}$.

Proof: The idea is quite simple. $M'$ simulates $M$ using a $\$$-shielded stack. When $M$'s computation is complete, $M'$ moves to a new stack-emptying vertex, stack-empty, at which $M'$ empties its stack of everything apart from the $\$$-shield. To then move to final$_{M'}$, $M'$ pops the $\$$, thus ensuring it has an empty stack when it reaches final$_{M'}$. $M'$ is illustrated in Figure 2. More precisely, $M'$ consists of the graph of $M$ plus three new vertices; start$_{M'}$, stack-empty, and final$_{M'}$. The following edges are also added: $(start_M, start_M)$ labeled Push $\$$, $\lambda$-edges from each of $M$'s final vertices to stack-empty, self-loops (stack-empty, stack-empty) labeled Pop $X$ for each $X \in \Gamma$, where $\Gamma$ is $M$'s stack alphabet (so $\$$ \neq X), and edge (stack-empty, final$_{M'}$) labeled Pop $\$.

It is clear that $L(M) = L(M')$. Nonetheless, we present the argument for completeness.
First, we show that $L(M') \subseteq L(M)$. Let $w \in L(M')$. Let $P'$ be a $w$-recognizing path in $M'$ and let $f$ be the final vertex of $M$ preceding stack-empty on the path $P'$. Removing the first edge in $P'$ and every edge including and after $(f, \text{stack-empty})$, leaves a path $P$ which is a $w$-recognizing path in $M$. Thus $L(M') \subseteq L(M)$.

Now we show $L(M) \subseteq L(M')$. Let $w \in L(M)$ and let $P$ be a $w$-recognizing path in $M$. Suppose that $P$ ends with string $s$ on the stack at final vertex $f$. We add the edges $(\text{start}_{M'}, \text{start}_M)$, $(f, \text{stack-empty})$, $|s|$ self-loops at stack-empty, and $(\text{stack-empty}, \text{final}_{M'})$ to $P$, yielding path $P'$ in $M'$. By choosing the self-loops to be labeled with the characters of $s^R$ in this order, we cause $P'$ to be a $w$-recognizing path in $M'$. Thus $L(M) \subseteq L(M')$.

\textbf{Lemma 3} Let $A$ and $B$ be languages recognized by PDAs $M_A$ and $M_B$, respectively, Then $A \circ B$ is also recognized by a PDA called $M_{A\circ B}$.

\textbf{Proof}: Let $M_A$ and $M_B$ be PDAs recognizing $A$ and $B$, respectively, where they each have just one final vertex that can be reached only with an empty stack.

$M_{A\circ B}$ consists of $M_A$, $M_B$ plus one $\lambda$-edge $(\text{final}_A, \text{start}_B)$. Its start vertex is start$_A$ and its final vertex is final$_B$. To see that $L(M_{A\circ B}) = A \circ B$ is straightforward. For $w \in L(M_{A\circ B})$ if and only if there is a $w$-recognizing path $P$ in $M_{A\circ B}$; $P$ is formed from a path $P_A$ in $M_A$ going from start$_A$ to final$_A$ (and which therefore ends with an empty stack), $\lambda$-edge $(\text{final}_A, \text{start}_B)$, and a path $P_B$ in $M_B$ going from start$_B$ to final$_B$. Let $u$ be the sequence of reads labeling $P_A$ and $v$ those labeling $P_B$. Then $w = uv$, $P_A$ is $u$-recognizing, and $P_B$ is $v$-recognizing (see Figure 3). So $w \in L(M_{A\circ B})$ if and only if $w = uv$, $u \in L(M_A) = A$ and $v \in L(M_B) = B$. In other words $w \in L(M_{A\circ B})$ if and only if $w \in A \circ B$.

\textbf{Lemma 4} Suppose that $L$ is recognized by PDA $M_L$ and suppose that $R$ is a regular language. Then $L \cap R$ is recognized by a PDA called $M_{L\cap R}$.

\textbf{Proof}: Let $M_L = (\Sigma, \Gamma_L, V_L, \text{start}_L, F_L, \delta_L)$ and let $R$ be recognized by DFA $M_R = (\Sigma, \Gamma_R, \text{start}_R, F_R, \delta_R)$. We will construct $M_{L\cap R}$. The vertices of $M_{L\cap R}$ will be 2-tuples, the first component corresponding to a vertex of $M_L$ and the second component
to a vertex of $M_R$. The computation of $M_{L \cap R}$, when looking at the first components along with the stack will be exactly the computation of $M_L$, and when looking at the second components, but without the stack, it will be exactly the computation of $M_R$. This leads to the following edges in $M_{L \cap R}$.

1. If $M_L$ has an edge $(u_L, v_L)$ with label Pop $A$, Read $b$, Push $C$ and $M_R$ has an edge $(u_R, v_R)$ with label $b$, then $M_{L \cap R}$ has an edge $((u_L, u_R), (v_L, v_R))$ with label Pop $A$, Read $b$, Push $C$.

2. If $M_L$ has an edge $(u_L, v_L)$ with label Pop $A$, Read $\lambda$, Push $C$ then $M_{L \cap R}$ has an edge $((u_L, u_R), (v_L, u_R))$ with label Pop $A$, Read $b$, Push $C$ for every $u_L \in V_R$.

The start vertex for $M_{L \cap R}$ is $(\text{start}_L, \text{start}_R)$ and its set of final vertices is $F_L \times F_R$, the pairs of final vertices, one from $M_L$ and one from $M_R$, respectively.

**Assertion.** $M_{L \cap R}$ can reach vertex $(v_L, v_R)$ on input $w$ if and only if $M_L$ can reach vertex $v_L$ and $M_R$ can reach vertex $v_R$ on input $w$.

Next, we argue that the assertion is true. For suppose that on input $w$, $M_{L \cap R}$ can reach vertex $(v_L, v_R)$ by computation path $P_{L \cap R}$. If we consider the first components of the vertices in $P_{L \cap R}$, we see that it is a computation path of $M_L$ on input $w$ reaching vertex $v_L$. Likewise, if we consider the second components of the vertices of $M_{L \cap R}$, we obtain a path $P'_R$. The only difficulty is that this path may contain repetitions of a vertex $u_R$ corresponding to reads of $\lambda$ by $M_{L \cap R}$. Eliminating such repetitions creates a path $P_R$ in $M_R$ reaching $v_R$ and having the same label $w$ as path $P'_R$.

Conversely, suppose that $M_L$ can reach $v_L$ by computation path $P_L$ and $M_R$ can reach $v_R$ by computation path $P_R$. Combining these paths, with care, gives a computation path $P$ which reaches $(v_L, v_R)$ on input $w$. We proceed as follows. The first vertex is $(\text{start}_L, \text{start}_R)$. Then we traverse $P_L$ and $P_R$ in tandem. Either the next edges in $P_L$ and $P_R$ are both labeled by a Read $b$ (simply a $b$ on $P_R$) in which case we use Rule (1) above to give the edge to add to $P$, or the next edge on $P_L$ is labeled by Read $\lambda$ (together with a Pop and a Push possibly) and then we use Rule (2) to give the edge to add to $P$. In the first case we advance one edge on both $P_L$ and $P_R$, in the second case we only advance on $P_L$. Clearly, the path ends at vertex $(v_L, v_R)$ on input $w$.

It is now easy to see that $L(M_{L \cap R}) = L \cap R$. For on input $w$, $M_{L \cap R}$ can reach a final vertex $v \in F = F_L \times F_R$ if and only if on input $w$, $M_L$ reaches a vertex $v_L \in F_L$ and $M_R$ reaches a vertex $v_R \in F_R$. That is, $w \in L(M_{L \cap R})$ if and only if $w \in L(M_L) = L$ and $w \in L(M_R) = R$, or in other words $w \in L(M_{L \cap R})$ if and only if $w \in L \cap R$.

\[\square\]