In the last two cases, the construction breaks down if either $\mathbf{M}_1$ or $\mathbf{M}_2$ is nondeterministic; the reader should be sure to understand just where the construction fails.

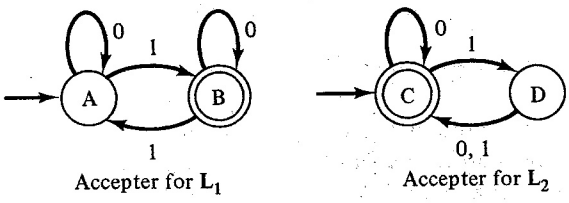
> **Example 5.18:** Figure 5.25 shows how two simple accepters for languages $\mathbf{L}_1$ and $\mathbf{L}_2$ may be used to derive accepters for $\mathbf{L}_1 \cup \mathbf{L}_2$, $\mathbf{L}_1 \cap \mathbf{L}_2$, and $\mathbf{L}_1 - \mathbf{L}_2$, according to the choice of accepting states in the state diagram for the parallel combination of the two accepters.

We now know that the class of finite-state languages is closed under the operations of intersection, difference, and complementation, as well as union, concatenation, and closure. Since every finite-state language is also a regular set, the class of finite-state languages is closed under reversal by the arguments of Section 5.3.1. We summarize these facts as a theorem.

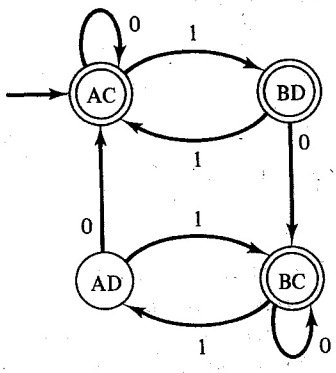
> **Theorem 5.7:** The class of finite-state languages is closed under the operations of set union, intersection, complementation, difference, concatenation, closure, and reversal. That is, if $\mathbf{L}_1$ and $\mathbf{L}_2$ are finite-state languages, then so are the following:
>
> 1. $\mathbf{L}_1 \cup \mathbf{L}_2$.
> 2. $\mathbf{L}_1 \cap \mathbf{L}_2$.
> 3. $\mathbf{L}_1^c$.
> 4. $\mathbf{L}_1 - \mathbf{L}_2$.
> 5. $\mathbf{L}_1 \cdot \mathbf{L}_2$.
> 6. $\mathbf{L}_1^*$.
> 7. $\mathbf{L}_1^R$.

It is important to understand the distinction between combining two machines in the manner just described and combining state diagrams as in the proof of Kleene's theorem. The manner of this section amounts to setting two physical, deterministic machines side by side and considering them as one; in the proof of Kleene's theorem, we consider two state diagrams as jointly representing the behavior of one machine. Combining state diagrams in the latter case has no convenient interpretation in terms of physical machines. In particular, the new state set is $\mathbf{Q}_1 \times \mathbf{Q}_2$ when physical machines are combined, but $\mathbf{Q}_1 \cup \mathbf{Q}_2$ when state diagrams are combined. For this reason, the machines that result from these methods are known as *product machines* and *sum machines*, respectively.

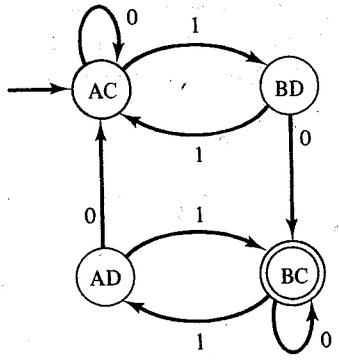
There is no simple way to directly combine nondeterministic accepters to obtain accepters for the intersection or difference of two finite-state languages. This is indicative of the difficulties we shall encounter in trying to apply these operations to context-free languages.



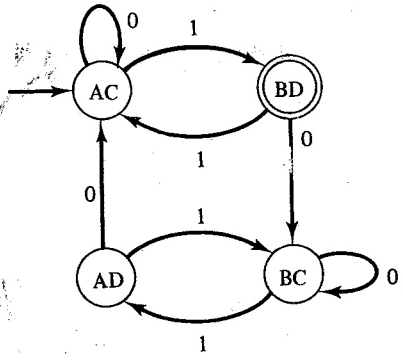
Accepter for L_1 | Accepter for L_2



Accepter for $L_1 \cup L_2$



Accepter for $L_1 \cap L_2$



Accepter for $L_1 - L_2$

Figure 5.25. Machines for Example 5.18.