DFA Operations

Complement, Intersection, Difference, and Equivalence of DFAs

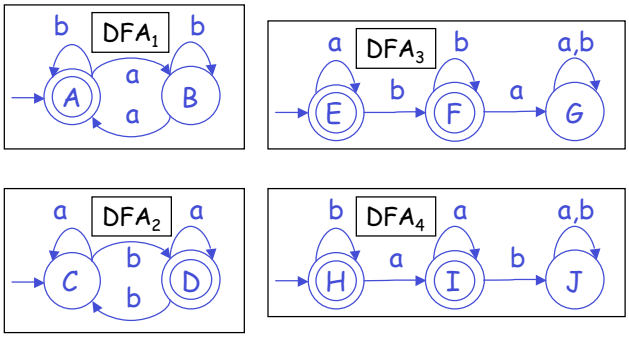
Thursday, October 11, 2007
Reading: Stoughton 3.11 – 3.12

CS235 Languages and Automata

Department of Computer Science
Wellesley College

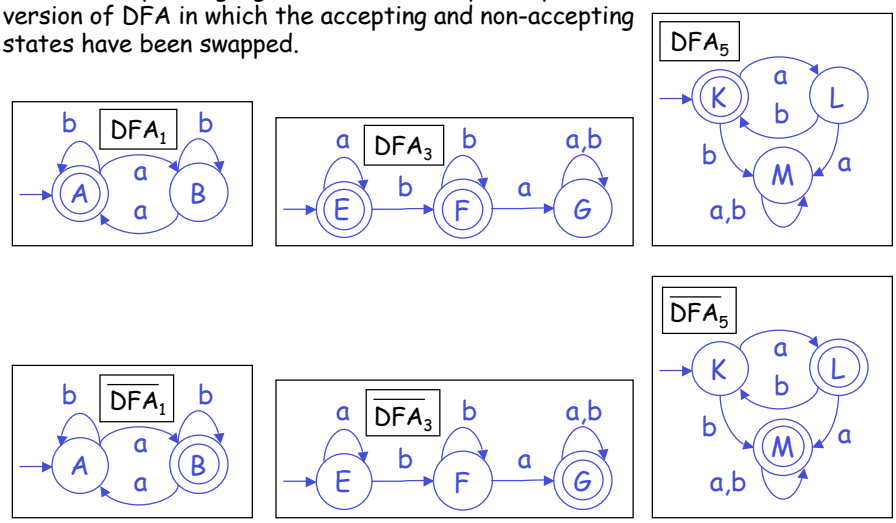
Some DFAs

Here are some simple DFAs we will use as examples in today's lecture. What languages do they accept?



Complement of DFAs

If DFA accepts language L, then $\overline{L}$ is accepted by $\overline{DFA}$, a version of DFA in which the accepting and non-accepting states have been swapped.

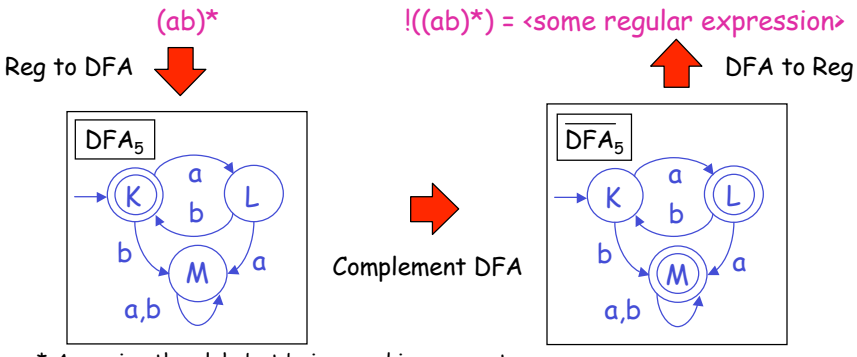


Complement of Regular Expressions

DFAs and regular expressions are two different ways of specifying regular languages.

Because DFAs can be complemented, regular languages are closed under complementation*.

So we could add a complement operator, !, to regular expressions.



* Assuming the alphabet being used is apparent.

Product of DFAs

There are several situations in which we want to run two DFAs in parallel on the same input. We can do this via the **product construction**.

Suppose $DFA_1 = (Q_1, s_1, A_1, T_1)$ and $DFA_2 = (Q_2, s_2, A_2, T_2)$

We define $DFA_1 \times DFA_2$ as follows:

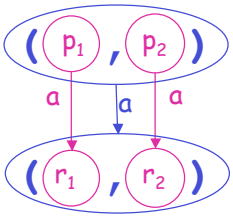
States: $Q_{1x2} = Q_1 \times Q_2$

Start State: $s_{1x2} = s_1 \times s_2$

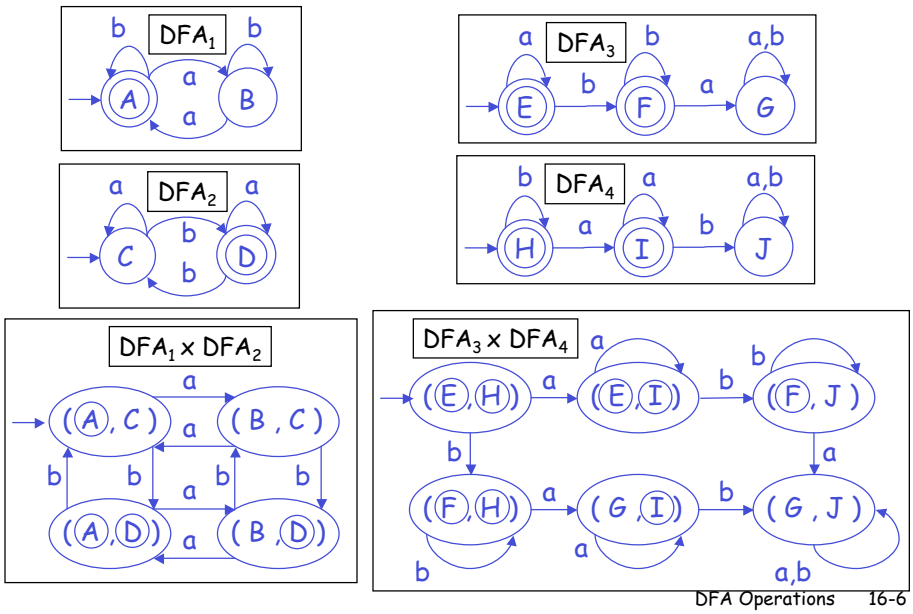
Accepting States: Definition depends on how we use product

Transitions:

$T_{1x2} = \{ ((p_1,p_2), a, (r_1,r_2)) \mid (p_1, a, r_1) \in T_1$ and $(p_2, a, r_2) \in T_2 \}$

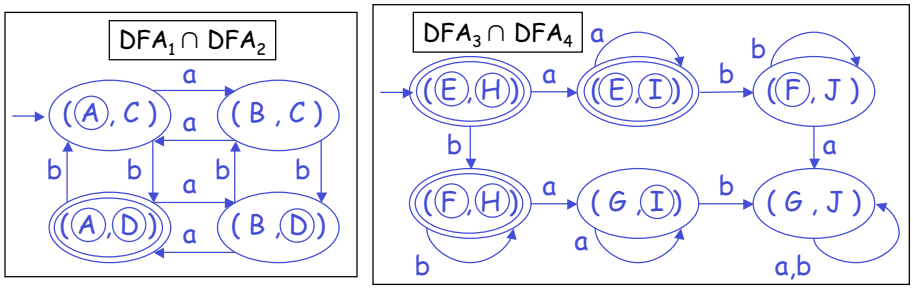


Sample Products



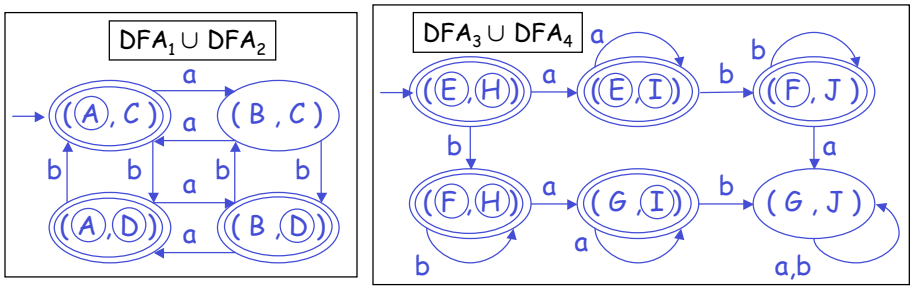
Intersection of DFAs

We can intersect DFA_1 and DFA_2 (written $DFA_1 \cap DFA_2$) by defining the accepting states of $DFA_1 \times DFA_2$ as those state pairs in which **both** states are accepting states of their DFAs.



Union of DFAs

We can union DFA_1 and DFA_2 (written $DFA_1 \cup DFA_2$) by defining the accepting states of $DFA_1 \times DFA_2$ as those state pairs in which **either** state is an accepting state of its DFA.



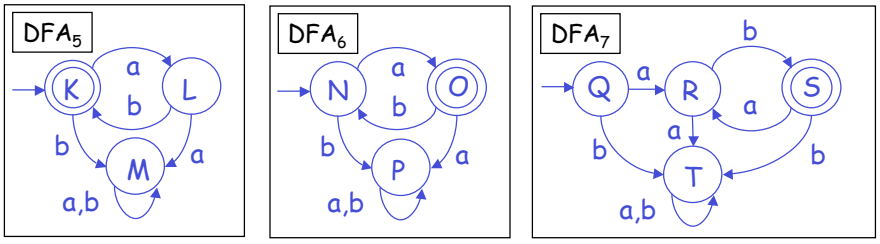
Some Consequences

- Regular languages are closed under union. We already knew that. But now we have an alternative way to union two FAs.
- Regular languages are closed under intersection. So we could add an intersection operator, &, to regular expressions. E.g., (a*b*) & (b*a*) would denote the same language as $DFA_3 \cap DFA_4$.
- We can define the difference of two DFAs in terms of complement and intersection:

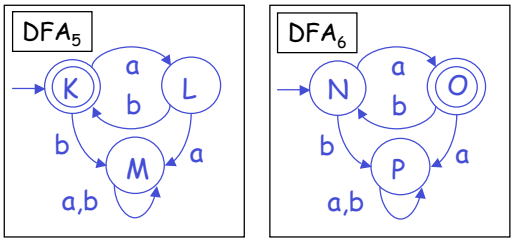
  $DFA_3 - DFA_4 = DFA_3 \cap DFA_4$

  So regular languages are closed under difference. And we could add a difference operator, -, to regular expressions. E.g., (a*b*) - (b*a*).

Are Any of the Following DFAs Equivalent?

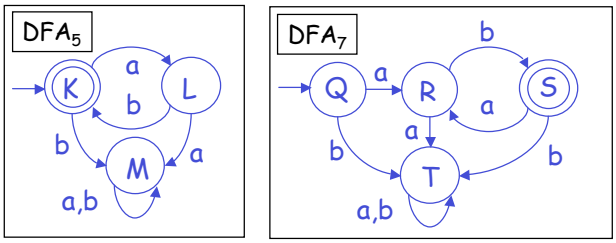


The Product of DFA_5 and DFA_6



$DFA_5 \times DFA_6$

The Product of DFA_5 and DFA_7



$DFA_5 \times DFA_7$

DFA Equivalence Algorithm

To determine if DFA_1 and DFA_2 are equivalent, construct $DFA_1 \times DFA_2$ and examine all state pairs containing at least one accepting state from DFA_1 or DFA_2:

If in all such pairs, both components are accepting, DFA_1 and DFA_2 are equivalent --- i.e., they accept the same language.

If in all such pairs, the first component is accepting but in some the second is not, the language of DFA_1 is a **superset** of the language of DFA_2 and it is easy to find a string accepted by DFA_1 and not by DFA_2

If in all such pairs, the second component is accepting but in some the first is not, the language of DFA_1 is a **subset** of the language of DFA_2, and it is easy to find a string accepted by DFA_2 and not by DFA_1

If none of the above cases holds, the languages of DFA_1 and DFA_2 are unrelated, and it is easy to find a string accepted by one and not the other.