Theorem 4.26: If $A$ is a DFA, and $M$ the DFA constructed from $A$ by the algorithm described in the statement of Theorem 4.24, then $M$ has as few states as any DFA equivalent to $A$. □

In fact we can say something even stronger than Theorem 4.26. There must be a one-to-one correspondence between the states of any other minimum-state $N$ and the DFA $M$. The reason is that we argued above how each state of $M$ must be equivalent to one state of $N$, and no state of $M$ can be equivalent to two states of $N$. We can similarly argue that no state of $N$ can be equivalent to two states of $M$, although each state of $N$ must be equivalent to one of $M$'s states. Thus, the minimum-state DFA equivalent to $A$ is unique except for a possible renaming of the states.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>→ A</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td>C</td>
<td>D</td>
<td>B</td>
</tr>
<tr>
<td>*D</td>
<td>D</td>
<td>A</td>
</tr>
<tr>
<td>E</td>
<td>D</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>G</td>
<td>E</td>
</tr>
<tr>
<td>G</td>
<td>F</td>
<td>G</td>
</tr>
<tr>
<td>H</td>
<td>G</td>
<td>D</td>
</tr>
</tbody>
</table>

Figure 4.14: A DFA to be minimized

4.4.5 Exercises for Section 4.4

* Exercise 4.4.1: In Fig. 4.14 is the transition table of a DFA.

  a) Draw the table of distinguishabilities for this automaton.

  b) Construct the minimum-state equivalent DFA.

Exercise 4.4.2: Repeat Exercise 4.4.1 for the DFA of Fig 4.15.
!! Exercise 4.4.3:!! Suppose that $p$ and $q$ are distinguishable states of a given DFA $A$ with $n$ states. As a function of $n$, what is the tightest upper bound on how long the shortest string that distinguishes $p$ from $q$ can be?

### 4.5 Summary of Chapter 4

- **The Pumping Lemma:** If a language is regular, then every sufficiently long string in the language has a nonempty substring that can be “pumped,” that is, repeated any number of times while the resulting strings are also in the language. This fact can be used to prove that many different languages are not regular.

- **Operations That Preserve the Property of Being a Regular Language:** There are many operations that, when applied to regular languages, yield a regular language as a result. Among these are union, concatenation, closure, intersection, complementation, difference, reversal, homomorphism (replacement of each symbol by an associated string), and inverse homomorphism.

- **Testing Emptiness of Regular Languages:** There is an algorithm that, given a representation of a regular language, such as an automaton or regular expression, tells whether or not the represented language is the empty set.

- **Testing Membership in a Regular Language:** There is an algorithm that, given a string and a representation of a regular language, tells whether or not the string is in the language.
\[ E \Rightarrow E \ast E \Rightarrow I \ast E \Rightarrow a \ast E \]  

Additionally, the derivation

\[ E \Rightarrow E \ast E \Rightarrow E \ast (E) \Rightarrow E \ast (E + E) \]  

shows that \( E \ast (E + E) \) is a right-sentential form. \( \square \)

5.1.7 Exercises for Section 5.1

Exercise 5.1.1: Design context-free grammars for the following languages:

* a) The set \( \{0^n1^n \mid n \geq 1\} \), that is, the set of all strings of one or more 0’s followed by an equal number of 1’s.

*! b) The set \( \{a^ib^jc^k \mid i \neq j \text{ or } j \neq k\} \), that is, the set of strings of a’s followed by b’s followed by c’s, such that there are either a different number of a’s and b’s or a different number of b’s and c’s, or both.
c) The set of all strings of a’s and b’s that are not of the form \(ww\), that is, not equal to any string repeated.

!! d) The set of all strings with twice as many 0’s as 1’s.

**Exercise 5.1.2:** The following grammar generates the language of regular expression \(0^*1(0 + 1)^*\):

\[
S \rightarrow A1B \\
A \rightarrow 0A | \epsilon \\
B \rightarrow 0B | 1B | \epsilon
\]

Give leftmost and rightmost derivations of the following strings:

* a) 00101.
  b) 1001.
  c) 00011.

**Exercise 5.1.3:** Show that every regular language is a context-free language. 
*Hint:* Construct a CFG by induction on the number of operators in the regular expression.

**Exercise 5.1.4:** A CFG is said to be right-linear if each production has at most one variable, and that variable is at the right end. That is, productions of a right-linear grammar are of the form \(A \rightarrow wB\) or \(A \rightarrow \epsilon\), where \(A\) and \(B\) are variables and \(w\) some string of zero or more terminals.

a) Show that every right-linear grammar generates a regular language. Construct an \(\epsilon\)-NFA that simulates leftmost derivations, using its states to represent the lone variable in the current left-sentential form.

b) Show that every regular language is generated by a right-linear grammar with a DFA.