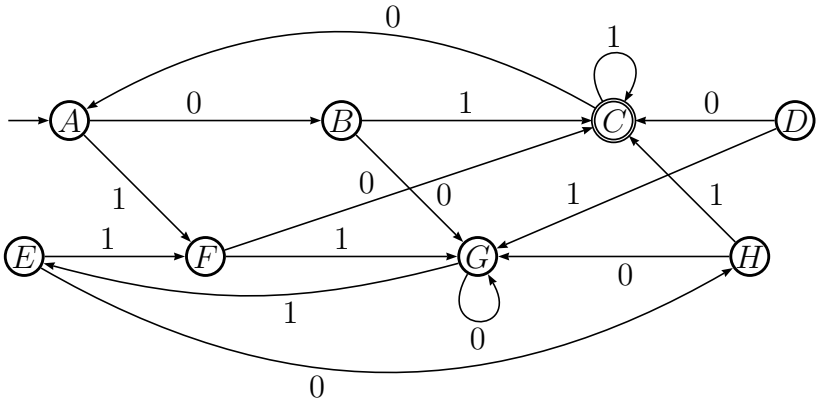
4. Done. Read off the equivalence classes: if (p, q) is not marked, then $p \approx q$.

Remark. We may have to revisit the same (p, q) pair several times, since combining two states can suddenly allow hitherto equivalent states to be markable.

Example 127. Minimize the following DFA



We start by setting up our table. We will be able to restrict our attention to the lower left triangle, since equivalence is symmetric. Also, each box on the diagonal will be marked with $\approx$, since every state is equivalent to itself. We also notice that state D is not reachable, so we will ignore it.

	A	B	C	D	E	F	G	H
A	$\approx$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
B		$\approx$	$-$	$-$	$-$	$-$	$-$	$-$
C			$\approx$	$-$	$-$	$-$	$-$	$-$
D	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
E				$-$	$\approx$	$-$	$-$	$-$
F				$-$		$\approx$	$-$	$-$
G				$-$			$\approx$	$-$
H				$-$				$\approx$

Now we split the states into final and non-final. Thus, a box indexed by p, q will be labelled with an X if p is a final state and q is not, or *vice versa*.

Thus we obtain

	A	B	C	D	E	F	G	H
A	$\approx$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
B		$\approx$	$-$	$-$	$-$	$-$	$-$	$-$
C	X_0	X_0	$\approx$	$-$	$-$	$-$	$-$	$-$
D	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
E			X_0	$-$	$\approx$	$-$	$-$	$-$
F			X_0	$-$		$\approx$	$-$	$-$
G			X_0	$-$			$\approx$	$-$
H			X_0	$-$				$\approx$

State C is inequivalent to all other states. Thus the row and column labelled by C get filled in with X_0. (We will subscript each X with the step at which it is inserted into the table.) However, note that C, C is not filled in, since $C \approx C$. Now we have the following pairs of states to consider:

$$\{AB, AE, AF, AG, AH, BE, BF, BG, BH, EF, EG, EH, FG, FH, GH\}$$

Now we introduce some notation which compactly captures how the machine transitions from a pair of states to another pair of states. The notation

$$p_1p_2 \xleftarrow{0} q_1q_2 \xrightarrow{1} r_1r_2$$

means $q_1 \xrightarrow{0} p_1$ and $q_2 \xrightarrow{0} p_2$ and $q_1 \xrightarrow{1} r_1$ and $q_2 \xrightarrow{1} r_2$. If one of p_1, p_2, r_1, or r_2 are already marked in the table, then there is a way to distinguish q_1 and q_2: they transition to inequivalent states. Therefore $q_1 \not\approx q_2$ and the box labelled by q_1q_2 will become marked. For example, if we take the state pair AB, we have

$$BG \xleftarrow{0} AB \xrightarrow{1} FC$$

and since FC is marked, AB becomes marked as well.

	A	B	C	D	E	F	G	H
A	$\approx$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
B	X_1	$\approx$	$-$	$-$	$-$	$-$	$-$	$-$
C	X_0	X_0	$\approx$	$-$	$-$	$-$	$-$	$-$
D	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
E			X_0	$-$	$\approx$	$-$	$-$	$-$
F			X_0	$-$		$\approx$	$-$	$-$
G			X_0	$-$			$\approx$	$-$
H			X_0	$-$				$\approx$

In a similar fashion, we examine the remaining unassigned pairs:

- $BH \xleftarrow{0} AE \xrightarrow{1} FF$. Unable to mark.
- $BC \xleftarrow{0} AF \xrightarrow{1} FG$. Mark, since BC is marked.
- $BG \xleftarrow{0} AG \xrightarrow{1} FE$. Unable to mark.
- $BG \xleftarrow{0} AH \xrightarrow{1} FC$. Mark, since FC is marked.
- $GH \xleftarrow{0} BE \xrightarrow{1} CF$. Mark, since CF is marked.
- $GC \xleftarrow{0} BF \xrightarrow{1} CG$. Mark, since CG is marked.
- $GG \xleftarrow{0} BG \xrightarrow{1} CE$. Mark, since CE is marked.
- $GG \xleftarrow{0} BH \xrightarrow{1} CC$. Unable to mark.
- $HC \xleftarrow{0} EF \xrightarrow{1} FG$. Mark, since CH is marked.
- $HG \xleftarrow{0} EG \xrightarrow{1} FE$. Unable to mark.
- $HG \xleftarrow{0} EH \xrightarrow{1} FC$. Mark, since CF is marked.
- $CG \xleftarrow{0} FG \xrightarrow{1} GE$. Mark, since CG is marked.
- $CG \xleftarrow{0} FH \xrightarrow{1} GC$. Mark, since CG is marked.
- $GG \xleftarrow{0} GH \xrightarrow{1} EC$. Mark, since EC is marked.

The resulting table is

	A	B	C	D	E	F	G	H
A	$\approx$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
B	X_1	$\approx$	$-$	$-$	$-$	$-$	$-$	$-$
C	X_0	X_0	$\approx$	$-$	$-$	$-$	$-$	$-$
D	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
E		X_1	X_0	$-$	$\approx$	$-$	$-$	$-$
F	X_1	X_1	X_0	$-$	X_1	$\approx$	$-$	$-$
G		X_1	X_0	$-$		X_1	$\approx$	$-$
H	X_1		X_0	$-$	X_1	X_1	X_1	$\approx$

Next round. The following pairs need to be considered:

$$\{AE, AG, BH, EG\}$$

The previously calculated transitions can be re-used; all that will have changed is whether the 'transitioned-to' states have been subsequently marked with an X_1:

AE: unable to mark

AG: mark because BG is now marked.

BH: unable to mark

EG: mark because HG is now marked

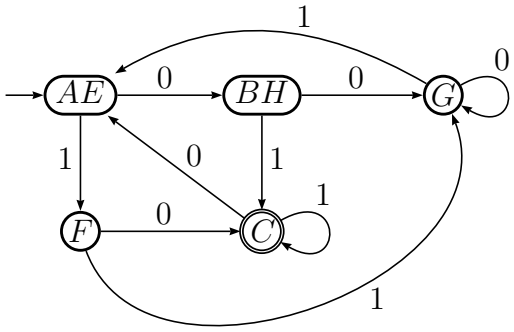
The resulting table is

	A	B	C	D	E	F	G	H
A	$\approx$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
B	X_1	$\approx$	$-$	$-$	$-$	$-$	$-$	$-$
C	X_0	X_0	$\approx$	$-$	$-$	$-$	$-$	$-$
D	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
E		X_1	X_0	$-$	$\approx$	$-$	$-$	$-$
F	X_1	X_1	X_0	$-$	X_1	$\approx$	$-$	$-$
G	X_2	X_1	X_0	$-$	X_2	X_1	$\approx$	$-$
H	X_1		X_0	$-$	X_1	X_1	X_1	$\approx$

Next round. The following pairs remain: $\{AE, BH\}$. However, neither makes a transition to a marked pair, so the round adds no new markings to the table. We are therefore done. The quotiented state set is

$$\{\{A, E\}, \{B, H\}, \{F\}, \{C\}, \{G\}\}$$

In other words, we have been able to merge states A and E, and B and H. The final automaton is given by the following diagram.



5.6 Decision Problems for Regular Languages

Now we will discuss some questions that can be asked about automata and regular expressions. These will tend to be from a general point of view, *i.e.*., involve arbitrary automata. A question that takes any automaton (or collection of automata) as input and asks for a terminating algorithm yielding a boolean (true or false) answer is called a *decision problem*, and a program that correctly solves such a problem is called a *decision algorithm*. Note well that a decision problem is typically a question about the (often infinite) set of strings that a machine must deal with; answers that involve running the machine on every string in the set are not useful, since they will take forever. That is not allowed: in every case, a decision algorithm must return a correct answer in finite time.

Here is a list of decision problems for automata and regular expressions:

1. Given a string x and a DFA M, $x \in \mathcal{L}(M)$?
2. Given a string x and an NFA N, $x \in \mathcal{L}(N)$?
3. Given a string x and a regular expression r, $x \in \mathcal{L}(r)$?
4. Given DFA M, $\mathcal{L}(M) = \emptyset$?
5. Given DFA M, $\mathcal{L}(M) = \Sigma^*$?
6. Given DFAs M_1 and M_2, $\mathcal{L}(M_1) \cap \mathcal{L}(M_2) = \emptyset$?