if M applied to w halts, and

$$q_0 w_M w \vdash^* y_1 q_n y_2,$$

if M applied to w does not halt. Here q_y and q_n are both final states of H.

Theorem 12.1

There does not exist any Turing machine H that behaves as required by Definition 12.1. The halting problem is therefore undecidable.

Proof: We assume the contrary, namely that there exists an algorithm, and consequently some Turing machine H, that solves the halting problem. The input to H will be the description (encoded in some form) of M, say w_M, as well as the input w. The requirement is then that, given any (w_M, w), the Turing machine H will halt with either a yes or no answer. We achieve this by asking that H halt in one of two corresponding final states, say, q_y or q_n. The situation can be visualized by a block diagram like Figure 12.1. The intent of this diagram is to indicate that, if M is started in state q_0 with input (w_M, w), it will eventually halt in state q_y or q_n. As required by Definition 12.1, we want H to operate according to the following rules:

$$q_0 w_M w \vdash_H^* x_1 q_y x_2,$$

if M applied to w halts, and

$$q_0 w_M w \vdash_H^* y_1 q_n y_2,$$

if M applied to w does not halt.

Figure 12.1

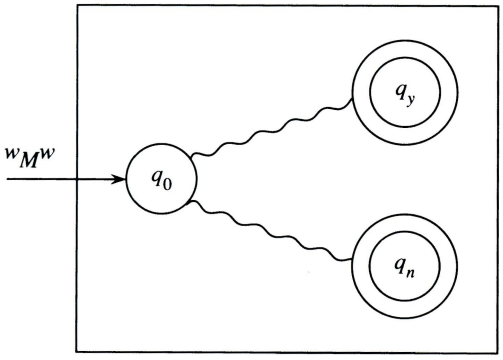
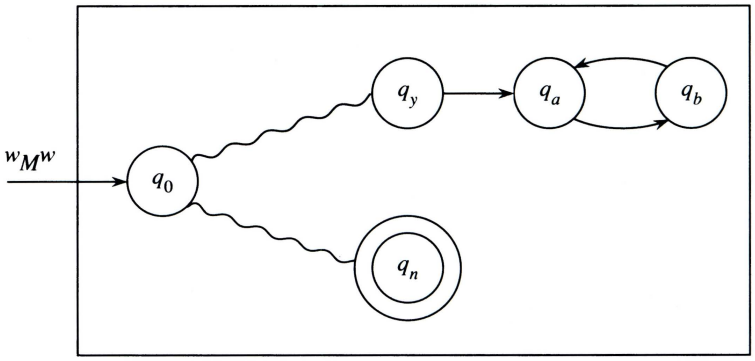


Figure 12.2



Next, we modify H to produce a Turing machine H' with the structure shown in Figure 12.2. With the added states in Figure 12.2 we want to convey that the transitions between state q_y and the new states q_a and q_b are to be made, regardless of the tape symbol, in such a way that the tape remains unchanged. The way this is done is straightforward. Comparing H and H' we see that, in situations where H reaches q_y and halts, the modified machine H' will enter an infinite loop. Formally, the action of H' is described by

$$q_0 w_M w \vdash^*_{H'} \infty,$$

if M applied to w halts, and

$$q_0 w_M w \vdash^*_{H'} y_1 q_n y_2,$$

if M applied to w does not halt.

From H' we construct another Turing machine $\hat{H}$. This new machine takes as input w_M, copies it, and then behaves exactly like H'. Then the action of $\hat{H}$ is such that

$$q_0 w_M \vdash^*_{\hat{H}} q_0 w_M w_M \vdash^*_{\hat{H}} \infty,$$

if M applied to w_M halts, and

$$q_0 w_M \vdash^*_{\hat{H}} q_0 w_M w_M \vdash^*_{\hat{H}} y_1 q_n y_2,$$

if M applied to w_M does not halt.

Now $\hat{H}$ is a Turing machine, so that it will have some description in Σ^*, say $\hat{w}$. This string, in addition to being the description of $\hat{H}$ can also be used as input string. We can therefore legitimately ask what would happen if $\hat{H}$ is applied to $\hat{w}$. From the above, identifying M with $\hat{H}$, we get

$$q_0\hat{w} \vdash^*_{\hat{H}} \infty,$$

if $\hat{H}$ applied to $\hat{w}$ halts, and

$$q_0\hat{w} \vdash^*_{\hat{H}} y_1 q_n y_2,$$

if $\hat{H}$ applied to $\hat{w}$ does not halt. This is clearly nonsense. The contradiction tells us that our assumption of the existence of H, and hence the assumption of the decidability of the halting problem, must be false. ■

One may object to Definition 12.1, since we required that, to solve the halting problem, H had to start and end in very specific configurations. It is, however, not hard to see that these somewhat arbitrarily chosen conditions play only a minor role in the argument, and that essentially the same reasoning could be used with any other starting and ending configurations. We have tied the problem to a specific definition for the sake of the discussion, but this does not affect the conclusion.

It is important to keep in mind what Theorem 12.1 says. It does not preclude solving the halting problem for specific cases; often we can tell by an analysis of M and w whether or not the Turing machine will halt. What the theorem says is that this cannot always be done; there is no algorithm that can make a correct decision for all w_M and w.

The arguments for proving Theorem 12.1 were given because they are classical and of historical interest. The conclusion of the theorem is actually implied in previous results as the following argument shows.

Theorem 12.2

If the halting problem were decidable, then every recursively enumerable language would be recursive. Consequently, the halting problem is undecidable.

Proof: To see this, let L be a recursively enumerable language on Σ, and let M be a Turing machine that accepts L. Let H be the Turing machine