We begin with the reversible but untidy computer mentioned earlier, which has produced, and failed to erase, a long history of its activity. Now, a tape full of random data cannot be erased except by an irreversible process; however, the history tape is not random—there exists a subtle mutual redundancy between it and the machine that produced it, which may be exploited to erase it reversibly. For example, if at the end of the computation a new stage of computation were begun using the inverse of the original transition function, the machine would begin carrying out the entire computation backward, eventually returning the history tape to its original blank condition[2]. Since the forward computation was deterministic and reversible, the backward stage would be also. Unfortunately, the backward stage would transform the output back into the original input, rendering the overall computation completely useless. Destruction of the desired output can be prevented simply by making an extra copy of it on a separate tape, after the forward stage, but before the backward stage. During this copying operation (which can be done reversibly if the tape used for the copy is initially blank), the recording of the history tape is suspended. The backward stage will then destroy only the original and not the copy. At the end of the computation, the computer will contain the (reconstructed) original input plus the intact copy of the output: all other storage will have been restored to its original blank condition. Even though no history remains, the computation is reversible and deterministic, because each of its stages has been so.

One disadvantage of the reversible machine would appear to be the large amount of temporary storage needed for the history—for a ν-step first stage, ν records of history would have to be written. In a later section it will be argued that by performing a job in many stages rather than just three, the required amount of temporary storage can often be greatly reduced. The final