Exercise 8.2.3: Design a Turing machine that takes as input a number \( N \) and a number \( w \). To be precise, the tape initially contains a $ followed by \( w \) and \( N \) in binary. The tape head is initially scanning the $ in state \( q_0 \). Your TM should halt with \( N+1 \), in binary, on its tape, scanning the leftmost symbol of \( N+1 \), in state \( q_f \). You may destroy the $ in creating \( N+1 \), if necessary. For instance, \( q_0 \$10011 \vdash q_f 10100 \), and \( q_0 \$11111 \vdash q_f 100000 \).

a) Give the transitions of your Turing machine, and explain the purpose of each state.

b) Show the sequence of ID's of your TM when given input $111$.

** Exercise 8.2.4: In this exercise we explore the equivalence between function computation and language recognition for Turing machines. For simplicity, we shall consider only functions from nonnegative integers to nonnegative integers, but the ideas of this problem apply to any computable functions. Here are the two central definitions:

- Define the graph of a function \( f \) to be the set of all strings of the form \( [x, f(x)] \), where \( x \) is a nonnegative integer in binary, and \( f(x) \) is the value of function \( f \) with argument \( x \), also written in binary.

- A Turing machine is said to compute function \( f \) if, started with any nonnegative integer \( x \) on its tape, in binary, it halts (in any state) with \( f(x) \), in binary, on its tape.

Answer the following, with informal, but clear constructions.

a) Show how, given a TM that computes \( f \), you can construct a TM that accepts the graph of \( f \) as a language.

b) Show how, given a TM that accepts the graph of \( f \), you can construct a TM that computes \( f \).

c) A function is said to be partial if it may be undefined for some arguments. If we extend the ideas of this exercise to partial functions, then we do not require that the TM computing \( f \) halts if its input \( x \) is one of the integers for which \( f(x) \) is not defined. Do your constructions for parts (a) and (b) work if the function \( f \) is partial? If not, explain how you could modify the construction to make it work.
Exercise 8.2.5: Consider the Turing machine

\[ M = (\{q_0, q_1, q_2, q_f\}, \{0, 1\}, \{0, 1, B\}, \delta, q_0, B, \{q_f\}) \]

Informally but clearly describe the language \( L(M) \) if \( \delta \) consists of the following sets of rules:

* a) \( \delta(q_0, 0) = (q_1, 1, R); \delta(q_1, 1) = (q_0, 0, R); \delta(q_1, B) = (q_f, B, R) \).

b) \( \delta(q_0, 0) = (q_0, B, R); \delta(q_0, 1) = (q_1, B, R); \delta(q_1, 1) = (q_1, B, R); \delta(q_1, B) = (q_f, B, R) \).

c) \( \delta(q_0, 0) = (q_1, 1, R); \delta(q_1, 1) = (q_2, 0, L); \delta(q_2, 1) = (q_0, 1, R); \delta(q_1, B) = (q_f, B, R) \).

8.3 Programming Techniques for Turing Machines

Our goal is to give you a sense of how a Turing machine can be used to compute in a manner not unlike that of a conventional computer. Eventually, we want to convince you that a TM is exactly as powerful as a conventional computer. In particular, we shall learn that the Turing machine can perform the sort of calculations on other Turing machines that we saw performed in Section 8.1.2 by a program that examined other programs. This “introspective” ability of both Turing machines and computer programs is what enables us to prove problems undecidable.

To make the ability of a TM clearer, we shall present a number of examples of how we might think of the tape and finite control of the Turing machine. None of these tricks extend the basic model of the TM; they are only notational conveniences. Later, we shall use them to simulate extended Turing-machine models that have additional features — for instance, more than one tape — by the basic TM model.

8.3.1 Storage in the State

We can use the finite control not only to represent a position in the “program” of the Turing machine, but to hold a finite amount of data. Figure 8.13 suggests this technique (as well as another idea: multiple tracks). There, we see the finite control consisting of not only a “control” state \( q \), but three data elements \( A \), \( B \), and \( C \). The technique requires no extension to the TM model; we merely think of the state as an address. In the case of Fig. 8.13, we should think of the state as being either with any one of three words.