Theory of Computation

Fifth macro: Y ← Y × X

Algorithm 17.8 simulates the macro Y ← Y × X in Simple Language. We can use the addition macro because integer multiplication can be simulated by repeated addition. Note that we need to preserve the value of X in a temporary variable, because in each addition we need the original value of X to be added to Y.

Algorithm 17.8 Macro Y ← Y × X

```
TEMP ← Y
Y ← 0
while (X)
{
        decr (X)
        Y ← Y + TEMP
}
```

Sixth macro: $Y \leftarrow Y^X$

Algorithm 17.9 simulates the macro $Y \leftarrow Y^X$ in Simple Language. We do this using the multiplication macro, because integer exponentiation can be simulated by repeated multiplication.

Algorithm 17.9 Macro $Y \leftarrow Y^X$

```
TEMP ← Y
Y ← 1
while (X)
{
        decr (X)
        Y ← Y × TEMP
}
```

Seventh macro: if X then A

Algorithm 17.10 simulates the seventh macro in Simple Language. This macro simulates the decision-making (if) statement of modern languages. In this macro, the variable X has only one of the two values 0 or 1. If the value of X is not 0, A is executed in the loop.

Algorithm 17.10 Macro if X then A

```
while (X)
{
        decr (X)
        A
}
```

Other macros

It is obvious that we need more macros to make Simple Language compatible with contemporary languages. Creating other macros is possible, although not trivial.

Input and output

In this simple language the statement read X can be simulated using ($X \leftarrow n$). We also simulate the output by assuming that the last variable used in a program holds what should be printed. Remember that this is not a practical language, it is merely designed to prove some theorems in computer science.

17-2 THE TURING MACHINE

The Turing machine was introduced in 1936 by Alan M. Turing to solve computable problems, and is the foundation of modern computers. In this section we introduce a very simplified version of the machine to show how it works.

Turing machine components

A Turing machine is made of three components: a tape, a controller and a read/write head (Figure 17.2).

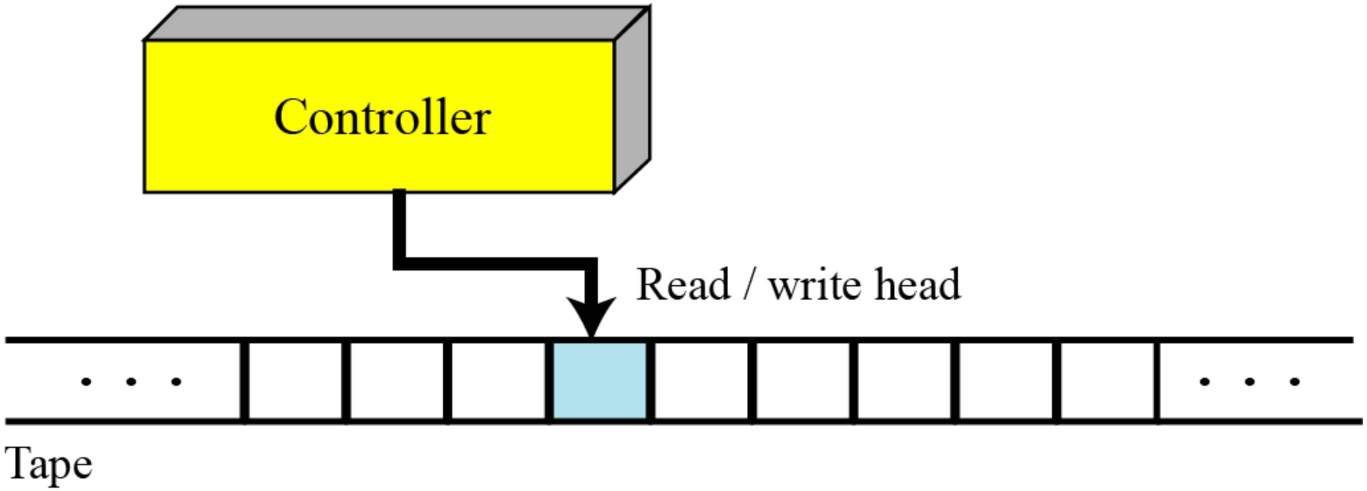

Figure 17.2 **The Turing machine**

Tape

Although modern computers use a random-access storage device with finite capacity, we assume that the Turing machine's memory is infinite. The tape, at any one time, holds a sequence of characters from the set of characters accepted by the machine. For our purpose, we assume that the machine can accept only two symbols: a blank (b) and digit 1.

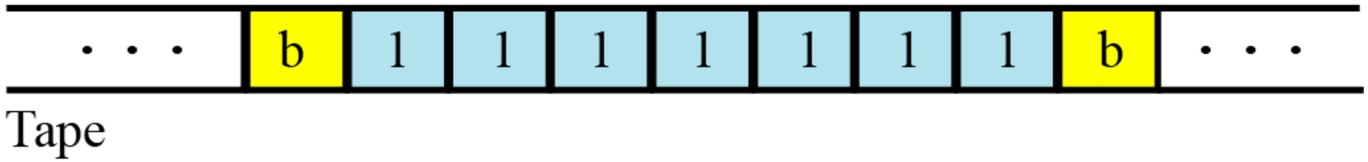

Figure 17.3 **The tape in the Turing machine**

Read/write head

The read/write head at any moment points to one symbol on the tape. We call this symbol the current symbol. The read/write head reads and writes one symbol at a time from the tape. After reading and writing, it moves to the left or to the right. Reading, writing and moving are all done under instructions from the controller.

Controller

The controller is the theoretical counterpart of the central processing unit (CPU) in modern computers. It is a finite state automaton, a machine that has a predetermined finite number of states and moves from one state to another based on the input.

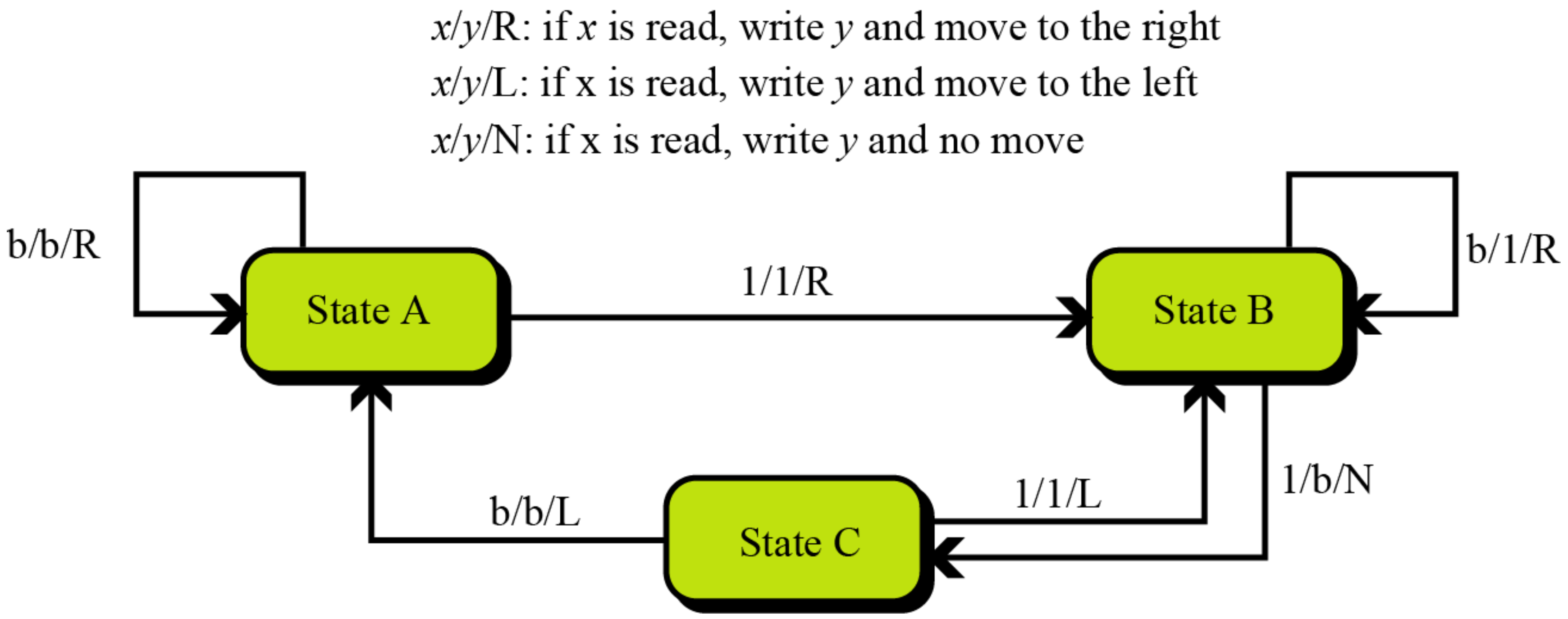

Figure 17.4 **Transition state diagram for the Turing machine**

Table 17.1 Transition table

Current state	*Read*	*Write*	*Move*	*New state*
A	b	b	R	A
A	1	1	R	B
B	b	1	R	B
B	1	b	N	C
C	b	b	L	A
C	1	1	L	B