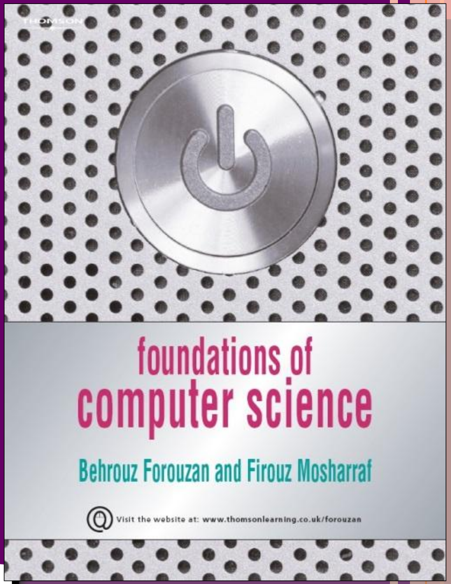

THOMSON
foundations of computer science
Behrouz Forouzan and Firouz Mosharraf
Visit the website at: www.thomsonlearning.co.uk/forouzan

Theory of Computation

Foundations of Computer Science

Objectives

After studying this chapter, the student should be able to:

- ❑ Describe a programming language we call Simple Language and define its basic statements.
- ❑ Write macros in Simple Language using the combination of simple statements.
- ❑ Describe the components of a Turing machine as a computation model.
- ❑ Show how simple statements in Simple Language can be simulated using a Turing machine.
- ❑ Understand the Church-Turing thesis and its implication.
- ❑ Define the Gödel number and its application.
- ❑ Understand the concept of the halting problem and how it can be proved that this problem is unsolvable.
- ❑ Distinguish between solvable and unsolvable problems.
- ❑ Distinguish between polynomial and non-polynomial solvable problems.

17-1 SIMPLE LANGUAGE

We can define a computer language with only three statements: the increment statement, the decrement statement and the loop statement (Figure 17.1).

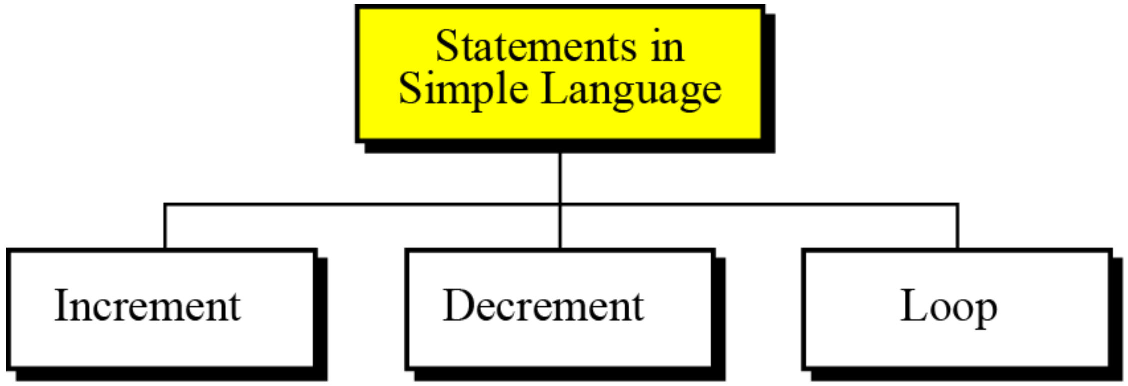


Figure 17.1 **Statements in Simple Language**

Increment statement

The increment statement adds 1 to a variable. The format is shown in Algorithm 17.1.

Algorithm 17.1 The increment statement

```
incr (X)
```

Decrement statement

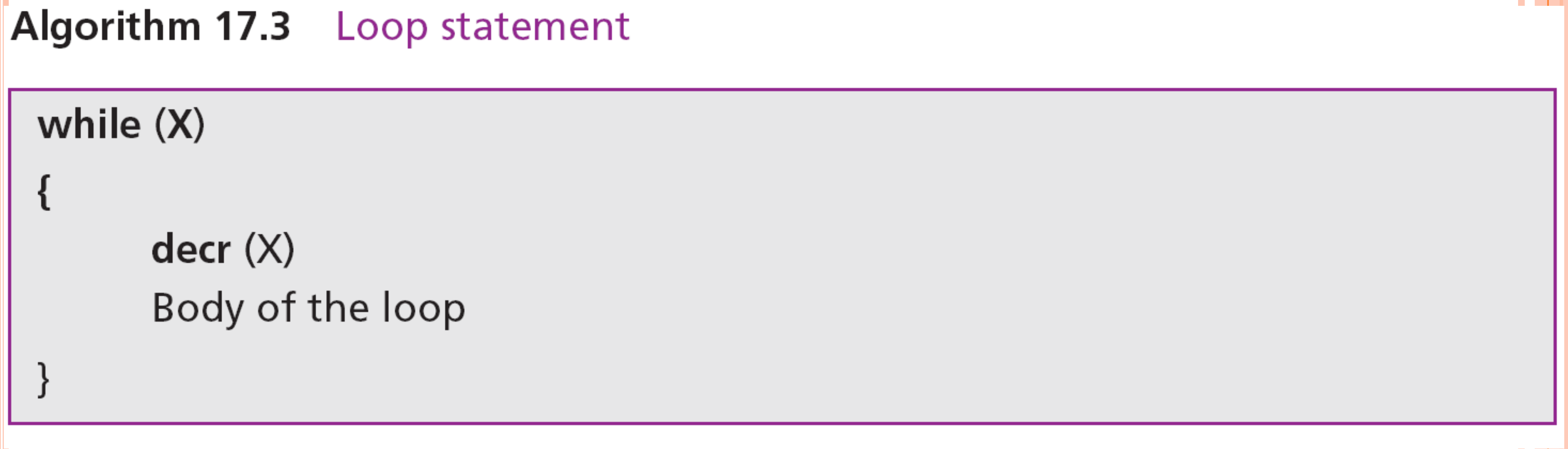
The decrement statement subtracts 1 from a variable. The format is shown in Algorithm 17.2.

Algorithm 17.2 The decrement statement

```
decr (X)
```

Loop statement

The loop statement repeats an action (or a series of actions) while the value of the variable is not 0. The format is shown in Algorithm 17.3.



Algorithm 17.3 Loop statement

```
while (X)
{
        decr (X)
        Body of the loop
}
```

The power of the Simple Language

It can be shown that this simple programming language with only three statements is as powerful—although not necessarily as efficient—as any sophisticated language in use today, such as C. To do so, we show how we can simulate several statements found in some popular languages.

Macros in Simple Language

We call each simulation a macro and use it in other simulations without the need to repeat code. A **macro** (short for macroinstruction) is an instruction in a high-level language that is equivalent to a specific set of one or more ordinary instructions in the same language.

First macro: X ← 0

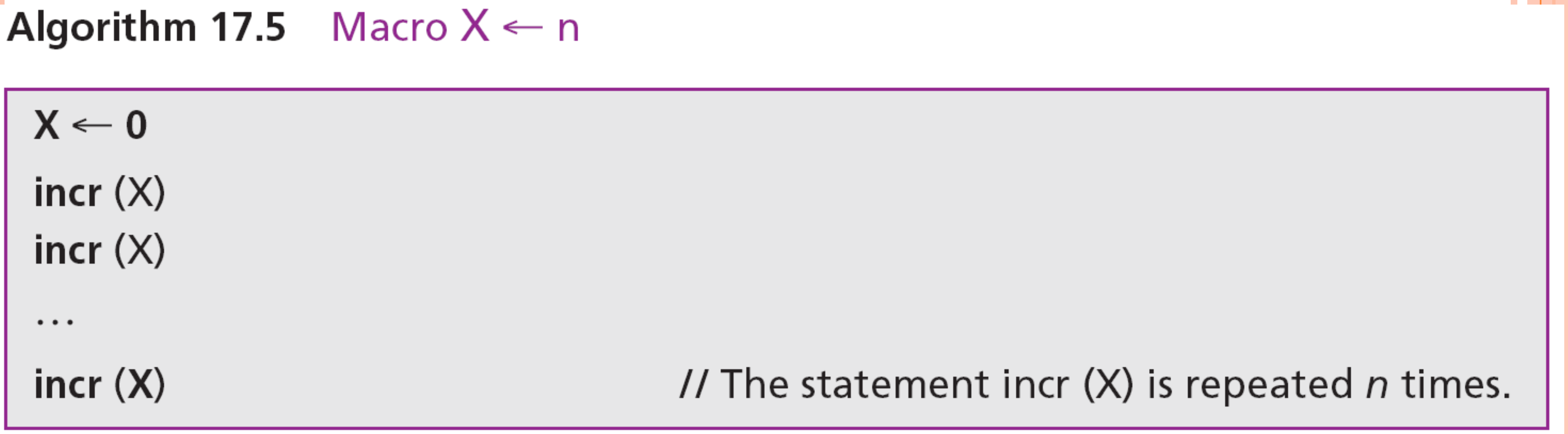
Algorithm 17.4 shows how to use the statements in Simple Language to assign 0 to a variable X. It is sometimes called clearing a variable.

Algorithm 17.4 Macro X ← 0

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

Second macro: $X \leftarrow n$

Algorithm 17.5 shows how to use the statements in Simple Language to assign a positive integer n to a variable X. First clear the variable X, then increment X *n* times.



Algorithm 17.5 Macro $X \leftarrow n$

```
X ← 0
incr (X)
incr (X)
...
incr (X)                                   // The statement incr (X) is repeated n times.
```

Third macro: Y ← X

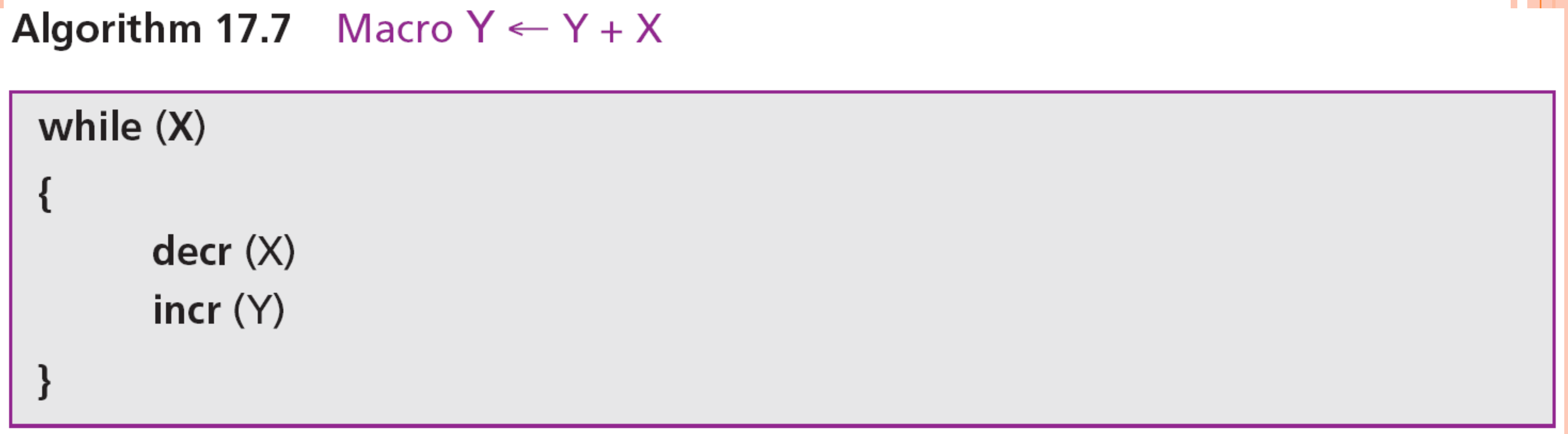
Algorithm 17.6 simulates the macro Y ← X in Simple Language. Note that we can use an extra line of code to restore the value of X.

Algorithm 17.6 Macro Y ← X

```
Y ← 0
while (X)
{
        decr (X)
        incr (Y)
}
```

Fourth macro: Y ← Y + X

Algorithm 17.7 simulates the macro Y ← Y + X in Simple Language. Again, we can use more code lines to restore the value of X to its original value.



Algorithm 17.7 Macro Y ← Y + X

```
while (X)
{
        decr (X)
        incr (Y)
}
```