• Function: A correspondence between a collection of possible input values and a collection of possible output values so that each possible input is assigned a single output.

• Computing a function: Determining the output value associated with a given set of input values.

• Noncomputable function: A function that cannot be computed by any algorithm.
**Figure 12.1** An attempt to display the function that converts measurements in yards into meters

<table>
<thead>
<tr>
<th>Yards (input)</th>
<th>Meters (output)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.9144</td>
</tr>
<tr>
<td>2</td>
<td>1.8288</td>
</tr>
<tr>
<td>3</td>
<td>2.7432</td>
</tr>
<tr>
<td>4</td>
<td>3.6576</td>
</tr>
<tr>
<td>5</td>
<td>4.5720</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Turing Machine Operation**

- Inputs at each step
  - State
  - Value at current tape position
- Actions at each step
  - Write a value at current tape position
  - Move read/write head
  - Change state

**Figure 12.2** The components of a Turing machine

- Control unit
- Tape
- Read/write head

**Figure 12.3** A Turing machine for incrementing a value

<table>
<thead>
<tr>
<th>Current state</th>
<th>Current cell content</th>
<th>Value to write</th>
<th>Direction to move</th>
<th>New state to enter</th>
</tr>
</thead>
<tbody>
<tr>
<td>START</td>
<td>+</td>
<td>*</td>
<td>Left</td>
<td>ADD</td>
</tr>
<tr>
<td>ADD</td>
<td>0</td>
<td>1</td>
<td>Right</td>
<td>RETURN</td>
</tr>
<tr>
<td>ADD</td>
<td>1</td>
<td>0</td>
<td>Left</td>
<td>CARRY</td>
</tr>
<tr>
<td>ADD</td>
<td>*</td>
<td>*</td>
<td>Right</td>
<td>HALT</td>
</tr>
<tr>
<td>CARRY</td>
<td>0</td>
<td>1</td>
<td>Right</td>
<td>RETURN</td>
</tr>
<tr>
<td>CARRY</td>
<td>1</td>
<td>0</td>
<td>Left</td>
<td>CARRY</td>
</tr>
<tr>
<td>CARRY</td>
<td>*</td>
<td>1</td>
<td>Left</td>
<td>OVERFLOW</td>
</tr>
<tr>
<td>OVERFLOW</td>
<td>*</td>
<td>*</td>
<td>Right</td>
<td>RETURN</td>
</tr>
<tr>
<td>RETURN</td>
<td>0</td>
<td>0</td>
<td>Right</td>
<td>RETURN</td>
</tr>
<tr>
<td>RETURN</td>
<td>1</td>
<td>1</td>
<td>Right</td>
<td>RETURN</td>
</tr>
<tr>
<td>RETURN</td>
<td>*</td>
<td>*</td>
<td>No move</td>
<td>HALT</td>
</tr>
</tbody>
</table>
Church-Turing Thesis

The functions that are computable by a Turing machine are exactly the functions that can be computed by any algorithmic means.

The Bare Bones Language

- Bare Bones is a simple, yet universal language.
- Statements
  - clear name;
  - incr name;
  - decr name;
  - while name not 0 do; ... end;

Universal Programming Language

A language with which a solution to any computable function can be expressed
- Examples: “Bare Bones” and most popular programming languages

Figure 12.4 A Bare Bones program for computing X x Y

clear Z;
while X not 0 do;
clear W;
while Y not 0 do;
  incr Z;
  incr W;
  decr Y;
end;
while W not 0 do;
  incr Y;
  decr W;
end;
decr X;
end;
clear Aux;
clear Tomorrow;
while Today not 0 do;
    incr Aux;
    decr Today;
end;
while Aux not 0 do;
    incr Today;
    incr Tomorrow;
    decr Aux;
end;

The Halting Problem

- Given the encoded version of any program, return 1 if the program is self-terminating, or 0 if the program is not.
Complexity of Problems

- **Time Complexity**: The number of instruction executions required
  - Unless otherwise noted, “complexity” means “time complexity.”
- A problem is in class \(O(f(n))\) if it can be solved by an algorithm in \(\Theta(f(n))\).
- A problem is in class \(\Theta(f(n))\) if the best algorithm to solve it is in class \(\Theta(f(n))\).

---

**Figure 12.8** A procedure `MergeLists` for merging two lists

```plaintext
procedure MergeLists (InputListA, InputListB, OutputList)
    if (both input lists are empty) then (Stop, with OutputList empty)
    if (InputListA is empty) then (Declare it to be exhausted)
        else (Declare its first entry to be its current entry)
    if (InputListB is empty) then (Declare it to be exhausted)
        else (Declare its first entry to be its current entry)
    while (neither input list is exhausted) do
        (Put the “smaller” current entry in OutputList:
            if (that current entry is the last entry in its corresponding input list)
                then (Declare that input list to be exhausted)
                else (Declare the next entry in that input list to be the list’s current entry)
        )
    Starting with the current entry in the input list that is not exhausted,
    copy the remaining entries to OutputList.
```

**Figure 12.9** The merge sort algorithm implemented as a procedure `MergeSort`

```plaintext
procedure MergeSort (List)
    if (List has more than one entry)
        then (Apply the procedure MergeSort to sort the first half of List;
            Apply the procedure MergeSort to sort the second half of List;
            Apply the procedure MergeLists to merge the first and second halves of List to produce a sorted version of List)
    
```

**Figure 12.10** The hierarchy of problems generated by the merge sort algorithm
- **Class P**: All problems in any class $\Theta(f(n))$, where $f(n)$ is a polynomial
- **Class NP**: All problems that can be solved by a nondeterministic algorithm in polynomial time
  - Nondeterministic algorithm = an “algorithm” whose steps may not be uniquely and completely determined by the process state
- Whether the class NP is bigger than class P is currently unknown.

**Key**: A value used to encrypt or decrypt a message
- **Public key**: Used to encrypt messages
- **Private key**: Used to decrypt messages

**RSA**: A popular public key cryptographic algorithm
- Relies on the (presumed) intractability of the problem of factoring large numbers
Encrypting the Message 10111

- Encrypting keys: \( n = 91 \) and \( e = 5 \)
- \( 10111_{\text{two}} = 23_{\text{ten}} \)
- \( 23^e = 23^5 = 6,436,343 \)
- \( 6,436,343 \div 91 \) has a remainder of 4
- \( 4_{\text{ten}} = 100_{\text{two}} \)
- Therefore, encrypted version of 10111 is 100.

Decrypting the Message 100

- Decrypting keys: \( d = 29 \), \( n = 91 \)
- \( 100_{\text{two}} = 4_{\text{ten}} \)
- \( 4^d = 4^{29} = 288,230,376,151,711,744 \)
- \( 288,230,376,151,711,744 \div 91 \) has a remainder of 23
- \( 23_{\text{ten}} = 10111_{\text{two}} \)
- Therefore, decrypted version of 100 is 10111.