Abstraction in Science, Mathematics, Software, and Markup

Roger L. Costello
March 2011

recur: happen or occur again. Synonym: **repeat**.

guise: general external appearance.

sameness: the quality of being alike.

Things recur over and over, albeit in different guises. **Abstraction** is seeing through the different guises to recognize what is recurring. Abstraction is recognizing sameness. Abstraction reduces cognitive load because experience in dealing with one thing can be applied to dealing with another.

In science, mathematics, software, and markup the same question applies:

Is something recurring?

Having discovered that something is recurring, scientists, mathematicians, software developers, and markup designers then proceed to specify it, characterize it, document it, represent it, formalize it, and/or create a symbol to represent it.

Abstraction in Science

Scientists look for recurring behaviors.

Example: A pencil or rock that is dropped from a building exhibit recurring behavior—they fall toward Earth at the same accelerating rate.

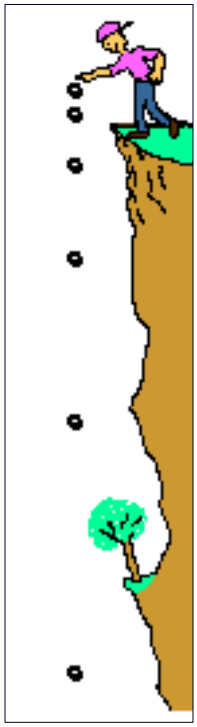


Figure 1: Objects fall toward Earth[1]

Despite having radically different guises, pencils and rocks exhibit the same behavior when dropped. This recurring behavior has been specified, characterized, documented, and formalized: Near the surface of the Earth, the rate of acceleration due to gravity is equal to 32 feet per second per second, or 9.8 meters per second per second[2].

The rate of fall may be represented using a table of values:

Time	Distance Travelled	Velocity
1 sec	16 feet	32 feet/sec
2 sec	64 feet	64 feet/sec
3 sec	144 feet	96 feet/sec
4 sec	256 feet	128 feet/sec
5 sec	400 feet	160 feet/sec

1 The picture is from: http://www.physicsclassroom.com/class/1dkin/u1l5a.cfm

2 http://wiki.answers.com/Q/What_is_the_rate_objects_fall

Alternatively, the rate of fall may be represented using equations:

- The distance (d) an object falls as a function of time (t) is: $d = gt^2/2$
- The velocity (v) of an object at a time (t) is: $v = gt$

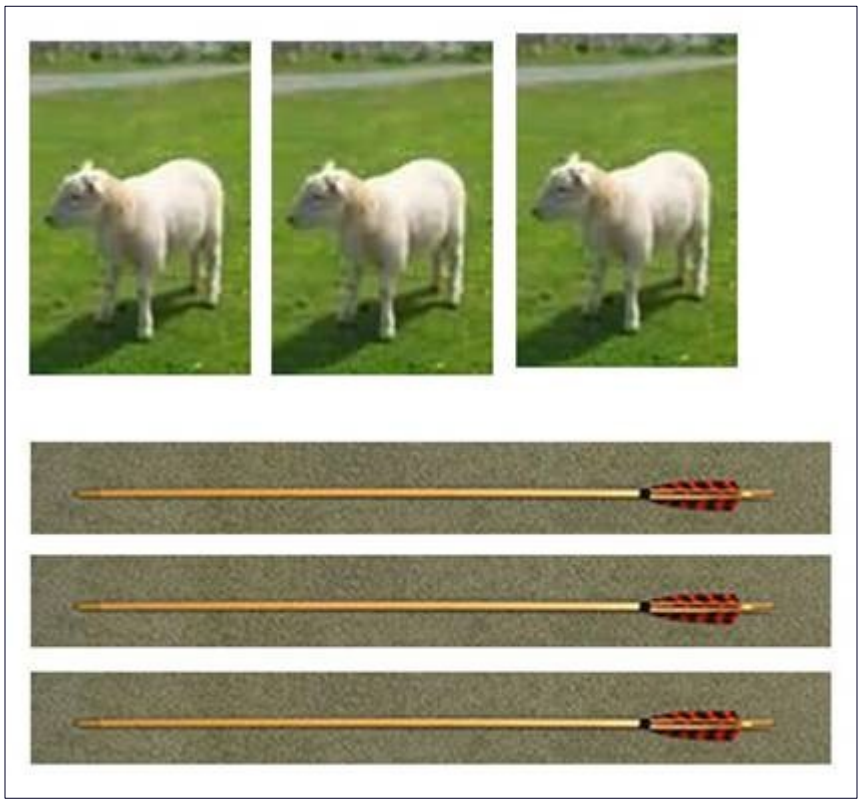
g = the acceleration due to gravity, 32 feet/sec^2 or 9.8 meters/sec^2

What is the best way to represent the data—as a table of values or as a formula? The best representation may depend on technology. With powerful handheld calculators it may require less space and even be faster to store the formula rather than a table of data. Prior to calculators it may have been more efficient in terms of time to carry around a table of values.

Abstraction in Mathematics

Mathematicians look for recurring properties.

Example: What do these sheep and arrows have in common?



.

In roughly 4000 B.C. the first civilizations learned to think about numbers as abstract concepts. They recognized that the sheep and arrows have something in common, a quantity called three,

which can be thought about independently of any physical objects. The appreciation of "number" as an abstract idea is a great, and perhaps the first, step in the founding of mathematics.

Abstraction in Software

Software developers look for recurring processes.

Example: Consider these two tasks:

Task #1: square each item in a list. E.g., square this list [2, 3, 4] to generate this list [4, 9, 16].

Taks #2: upper-case each character in a string. E.g., upper-case this string "Hello World" to generate this string "HELLO WORLD".

Both tasks involve producing new lists that are the same lengths as their input lists, and performing only one piece of work per item in the list. This task recurs frequently, particularly in functional programming languages. *It would be good to abstract out that common pattern so that it can be reused with less boilerplate.* Most functional programming languages provide a function, `map` which implements that common pattern. `map` takes a function and applies it to every element of a list, returning a new list constructed from the results of these applications.

Abstraction in Markup

Markup designers look for recurring sets of markup and recurring markup patterns.

Example: it is observed that all appliances have a model number, a description, and a warranty. This recurring set of markup can be captured in a schema language such as XML Schemas. Here is how that abstraction is expressed:

```
<xsd:complexType name="appliance">
    <xsd:sequence>
        <xsd:element name="model-number" type="xsd:ID"/>
        <xsd:element name="description" type="xsd:string"/>
        <xsd:element name="warranty" type="xsd:string"/>
    </xsd:sequence>
</xsd:complexType>
```

That abstraction identifies specific recurring elements, but sometimes it is not specific elements that recur, but rather patterns of markup.

Example: it may be observed that this markup pattern recurs frequently:

```
<element attribute="value">simple value</element>
```

Here are a couple instances of that markup pattern:

```
<altitude units="feet">12000</altitude>

<cost currency="USD">49.99</cost>
```

This recurring markup pattern can (I think) be captured in Relax NG. Here is how that abstraction is expressed:

```
<element>
    <anyName ns=" "/>
    <attribute>
        <anyName ns=" "/>
        <text/>
    </attribute>
    <text/>
</element>
```