Why is a Function Defined as a Set of Ordered Pairs?

Ann Q. Gates and **Vladik Kreinovich**
Department of Computer Science
The University of Texas at El Paso
El Paso, Texas 79968
{agates, vladik}@cs.utep.edu

Abstract

In this short note, we describe the reasoning that makes the standard mathematical definition of a function (as a set of ordered pairs) more natural for students in an introduction to computer science course.

Standard Definition of a Function

The standard definition of a function is not 100% intuitive. The introductory course in computer science typically reviews the basic mathematical definitions and relates them to concepts in computer science (see, e.g., [1]). One of these definitions, a *function f* from the set X to the set Y, is defined as a set of ordered pairs (x,y) such that for every $x \in X$ there exists one and only one $y \in Y$ for which $(x,y) \in f$.

Computer science students usually know this definition from prerequisite mathematics classes. Mathematics and computer science textbooks usually explain this definition using an example of a function from real numbers (or integers) to real numbers, where the pairs $(x,(x))$ form a *graph*. However, students often do not feel comfortable with this definition, especially when it is applied to input data that is more complicated than real numbers or integers. For instance, consider boolean functions *not* and, similarly, *and*, that can be defined as {(true,false), (false,true)} and {((true,true),true), ((true,false),false), ((false,true),false), ((false,false),false)}, respectively.

In this note, we describe an additional explanation that makes this definition more intuitive for our students. We hope that this explanation will be useful to other teachers of the computer science introductory course.

An Intuitive Explanation

In computer science we are mainly interested in the functions that can be actually *computed*; that is, functions for which there exists an algorithm that transforms an arbitrary input $x \in X$ into an output $f(x)$. In other words, a function can be given as a program, a program function, or a program procedure.

In some cases, we know the *algorithm* that computes $f(x)$ from x. However, in many cases we do not know the algorithm (e.g., consider buying software from a company). The program works as a *black box* illustrated as follows.

- We input x,
- Something occurs, but we do not know what, and
- $f(x)$ appears.

To describe each computation, we can *record* what we see:
- Each input value, x, and
- The corresponding output value, $f(x)$.

The *diary* (or record) of the computation process then contains two entries: x and $f(x)$, with x being the first and $f(x)$ the second. Thus, the diary representing each computation constitutes an *ordered pair*, $(x, f(x))$, of entries. Suppose a computation takes a string from the set {apple, atoyota, eve, axle, pop} and returns *true* if the string is a palindrome and *false*, otherwise. Then the diary will record the following: (apple,false), (atoyota,true), (eve,true), (axle,false), (pop,true).

Because the function can be applied to different values $x \in X$ to describe the function, we must describe *all possible* computations; that is, all possible pairs $(x,f(x))$. When we design a function, we do not know which inputs x will be used first and which inputs will be used later. Because there is no specific order between the different pairs (x,f(x)), these pairs form a *set*. Indeed, from the computer science viewpoint, it is natural to represent a function f as a *set* of *ordered pairs* $(x,f(x))$.

Acknowledgments: This work was supported in part by the NSF under grant No. EEC-9322370 and by NASA under cooperative agreement NCCW-009.

Reference

[1] Bernat, A., W.J. Bradley, R.D. Cupper, and G.W. Scragg, *Fundamentals of Computing I*, McGraw Hill, New York, 1994.