Tutorial #8. Functions

General info

Function is a relation between 2 sets. More exactly, it describes relation between elements of input set (domain) and output set (codomain) such that each **domain member is related exactly to single output** member (! and that's all !). This relation is denoted as F: X→Y, where F – function name, X – is an input set (domain), Y – output set (codomain). Domain elements sometimes referred as **function arguments**. Domain elements – exactly all elements that can be passed to a function; codomain – a set, containing a subset with all output values (codomain can contain additional elements).

E.g. $F(x)=x^2$: R→R has domain equals to set real numbers, and codomain **R**, although it does not contain negative elements. Codomain F[X] is called **set image** *iff* it contains elements F(x) produced by function F for all possible domain X members. Image is denoted by square brackets.

For function F: X→Y, F[X] ⊆Y. There's also reverse operation: finding preimage by **image set**. Operation of finding preimage is what usually done when you are bruteforcing password given a hash values (image).

Task: find preimage: $\sin^{-1}$[{0.7017, 0.52742}]

Solution: $\{\forall k \in Z |\ \pm\arcsin(0.7017) + 2k\pi, \pm\arcsin(0.52742) + 2k\pi\} =$
$\left\{\forall k \in Z \middle| \pm\frac{7}{9} + 2k\pi, \pm\frac{5}{9} + 2k\pi\right\}$

Defining function

Function can be **described** in multiple ways.

1) We can describe function explicitly using ordered pairs, where first element is from domain set, and second corresponding codomain member: (x, F(x)). Set of all possible pairs is called **graph of a function**. For example, logical negation ~:B→B can be easily explicitly described as **~ == {(0, 1), (1, 0)}.** You can draw arrow diagrams for explicit graphs (TAs, draw few arrow diagrams, they'll be in hometask). If both domain and codomain are real, ordered pairs are called **Cartesian coordinates.**
2) We can use operations, combined into **formulas**, allowed on our domain set for building a graph set. E.g. arithmetic is allowed on a set of real numbers, k-th derivative is allowed on a set of smooth functions, "translation to Russian" is allowed on a set of English words, etc. Moreover, function description can combine multiple formulas into **algorithms**. E.g. we can say "first multiply by 2, then add 1" to describe a function for building odd numbers.
3) We can use piecewise definition in cases, when it is hard to enumerate elements:

$$sign(x) = \begin{cases} 1, x > 0 \ \mid same\ as\ \forall x > 0, (x, 1) \\ -1, x < 0 \mid same\ as\ \forall x < 0, (x, -1) \\ 0, x = 0 \mid same\ as\ (0, 0) \end{cases}$$

Function can accept multiple attributes. To unify approach of describing functions, we can use **Cartesian product** of input sets to find domain – these are all possible tuples of input set members. Cartesian product is calculated when you implement **cross join** (inner join without constraint) in SQL statement.

```
SELECT * FROM Person, City
```

Important property of Cartesian product is |A*B|=|A|*|B|. That means that if you write unconstrained SQL join for two 1000-record tables, your statement would have to process 1M lines.

Injective, surjective and bijective functions

Injective function mean that your function never produce similar output for different domain values. E.g. *x^3, $\log_2 x$, inverse(word)* are injective. *x^2, countLetters(word)* are non-injective (TAs: ask students to guess).

1) Are hash functions injective? (no, that's the main purpose of hash function)
2) Prove or disprove that F(x) = (3 x + 2) % 11: [0..10]→Z is injective. (Either code or modular arithmetic). F(x):[0..11] →Z?

Surjective functions are functions where codomain is exactly an image of the domain – each element in codomain has at least one related element in a domain.

Non-injective and surjective:

- $\mathbf{R} \to \mathbf{R} : x \mapsto (x-1)x(x+1) = x^3 - x$
- $\mathbf{R} \to [-1,1] : x \mapsto \sin(x)$

Bijective functions are both injective and surjective.

- For every set A the **identity function id$_A$** and thus specifically $\mathbf{R} \to \mathbf{R} : x \mapsto x$.
- $\mathbf{R}^+ \to \mathbf{R}^+ : x \mapsto x^2$ and thus also its inverse $\mathbf{R}^+ \to \mathbf{R}^+ : x \mapsto \sqrt{x}$.
- $\exp : \mathbf{R} \to \mathbf{R}^+ : x \mapsto e^x$, $\ln : \mathbf{R}^+ \to \mathbf{R} : x \mapsto \ln x$

Function composition

We often use function composition, when write something like this: $\sin(x^2)$ or this e^{x+y}. The former can be generalized to *$t(x)=g(f(x))$*, the latter to *$t(x,y)=g(f(x,y))$*. This relation is called composition, and if you want to write it without variables, in will be *$t=g \circ f$*. For composition it is very important, that co-domain of ***f*** is a subset of domain of **g**.

Try the following example in your browser consoles (F12) together with students. Refresh that in JavaScript functions are first-class citizens and can be stored in variables. While typing, ask them to give definition to functions: g:R→R, f:R→R, i: words→words, h: words→Z^+

```
g = function(x) { return x*x; }
f = function(x) { return Math.sqrt(x); }
f(g(100))
g(f(100))
// after these lines ask about the difference and ask to write
formulas at the whiteboard
h = function(x) { return x.split('').reverse().join(''); }
i = function(x) { return x.length; }
h('word')
g(h('word'))
// what's going wrong here? – domain of g and codomain of h don't
match
g(i(h('word')))
// why it works now? – Z+ is a subset of R, so it's ok
```

Let's return to our functions g() and f(). Write

```
t1 = function(x) { return g(f(x)); }
t2 = function(x) { return f(g(x)); }
// try these functions t1(x) and t2(x) with different params. Why
they are giving the same result? Why this result is equal to
param?
```

There special operation, called **inverse of the function**. This operation "switches sides" in ordered pairs: *f:X→Y ; f^{-1}:Y→X,* defined by $\operatorname{graph} f^{-1} = \{(y,x) \mid y = f(x)\}$. If we want $f^{-1}(x)$ be a function, we have to ensure, f is bijective.

Ask students, why f(x) cannot be

- Only injective
- Only surjective?

If we reduce g(x) and f(x) domains to R+, then we *$g=f^{-1}$* and vice versa.

If *f* is an **invertible function** with domain *X* and range *Y*, then

- $f^{-1}\left(f(x)\right) = x$, for every $x \in X$.
- Using the composition of functions we can rewrite this statement as follows:
- $f^{-1} \circ f = \mathrm{id}_X$,

Knowing that $F = f(C) = \frac{9}{5}C + 32$; implement F in any programming language. Evaluate and implement $F^{-1}(C) = C(F)$. Check combination is giving you identity function.