# Lively4 Blackbox

Jonas Chromik, Christopher Weyand

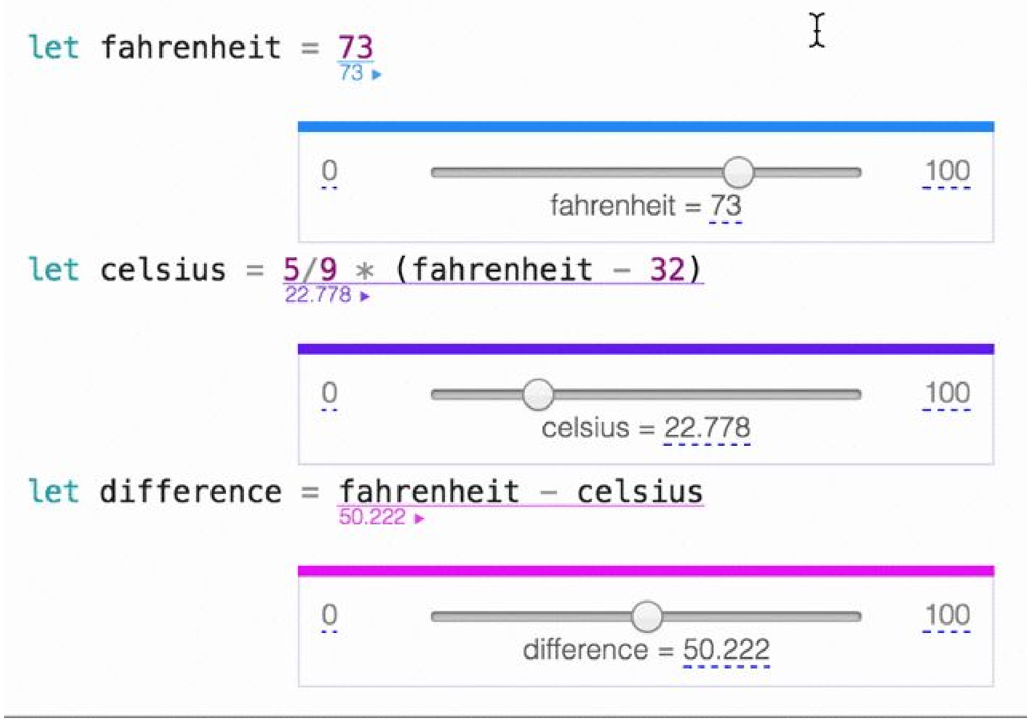
Software Design Seminar

Winter Term 2016/2017

# Reference: Carbide

Aims to be a “new kind of **programming environment**”. [1]

Input and output of a script can be modified **bi-directionally**.



```
let fahrenheit = 73
                 73 ▸
```

fahrenheit = 73

```
let celsius = 5/9 * (fahrenheit - 32)
              22.778 ▸
```

celsius = 22.778

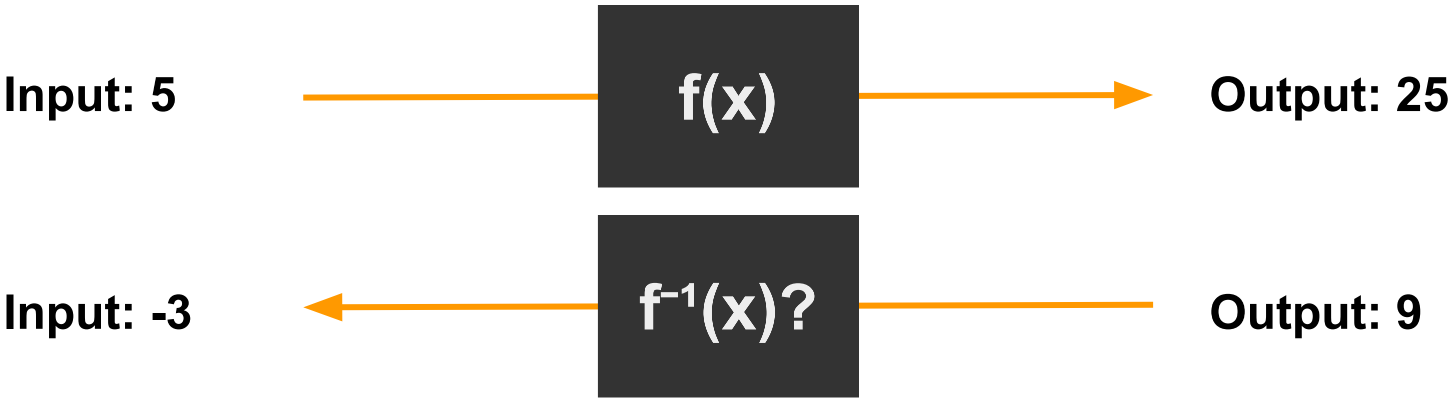
```
let difference = fahrenheit - celsius
                 50.222 ▸
```

difference = 50.222

# Problem

# Execute Functions! But backwards?

$f(x) = x * x$



Input: 5

$f(x)$

Output: 25

Input: -3

$f^{-1}(x)$?

Output: 9

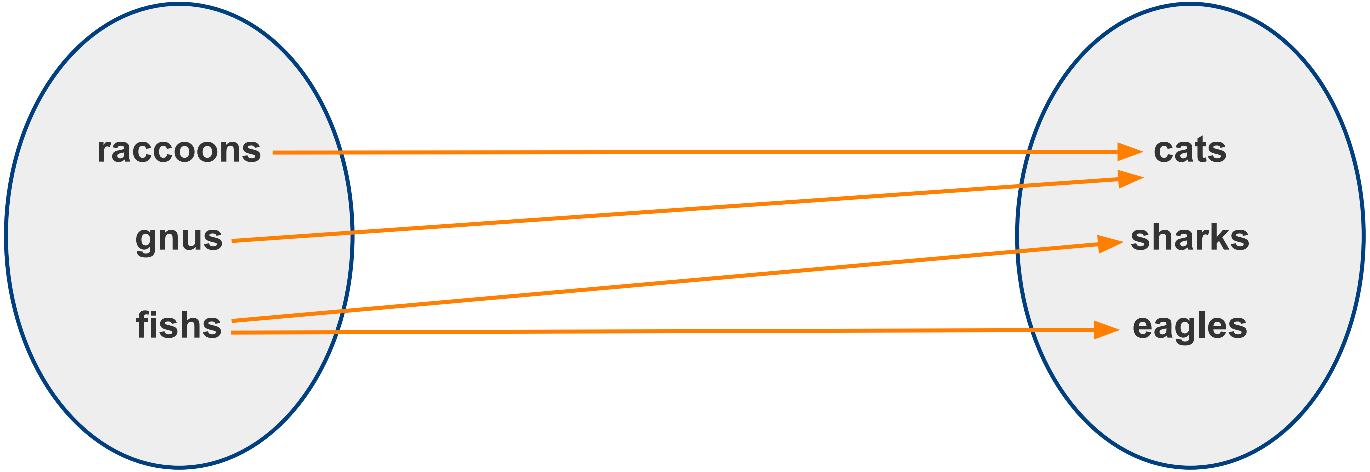
# Many-to-Many Relationships

```
function transformation(input) {
    var output = {};

    output.cats = input.raccoons * input.gnus;
    output.sharks = input.fishs + “_big”;
    output.eagles = input.fishs.length;

    return output;
}
```

# Many-to-Many Relationships



# Problems with $f^{-1}$

No solutions?

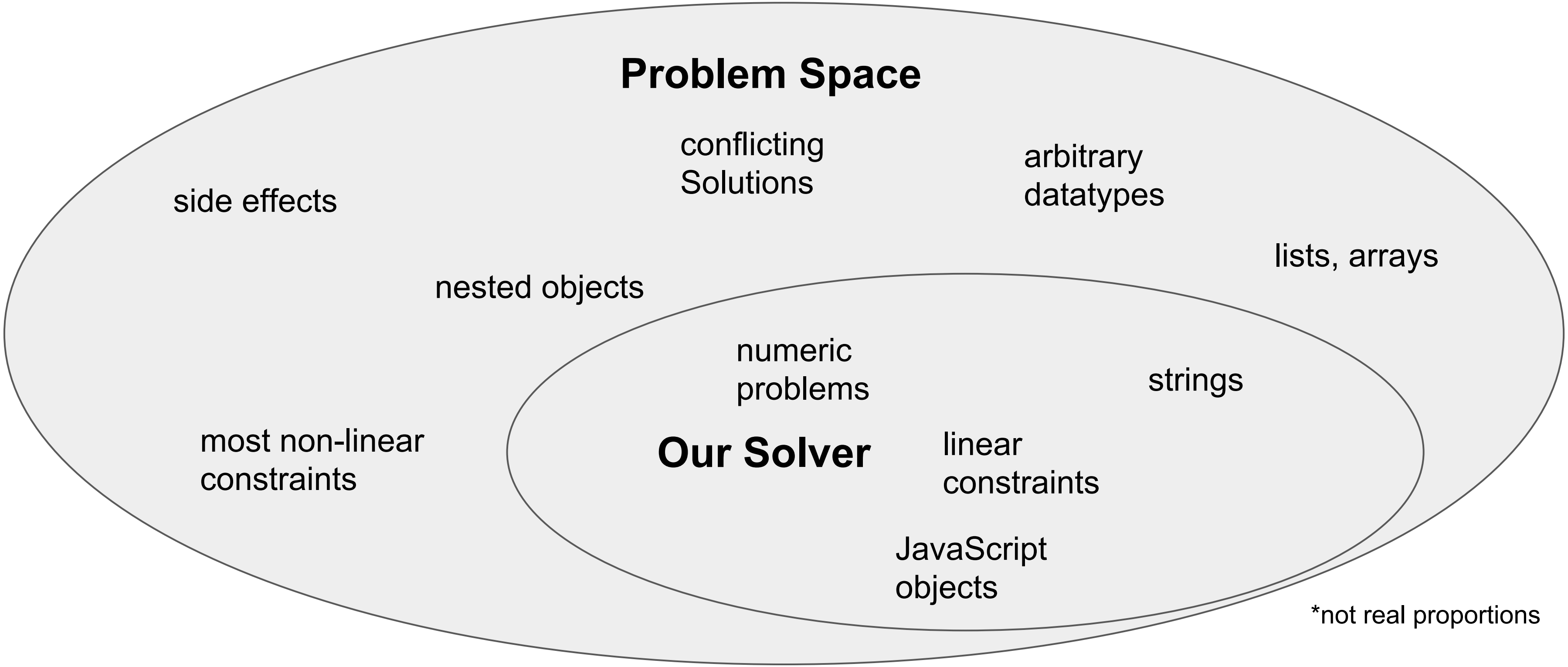
Many solutions?

Strings, objects, arrays, …?

Side effects?

Many values: $f(x1, x2, ...) = y$ $f(x1, x2, \ldots) = (y1, y2, \ldots)$

# Big Picture!



*not real proportions

# General Approach

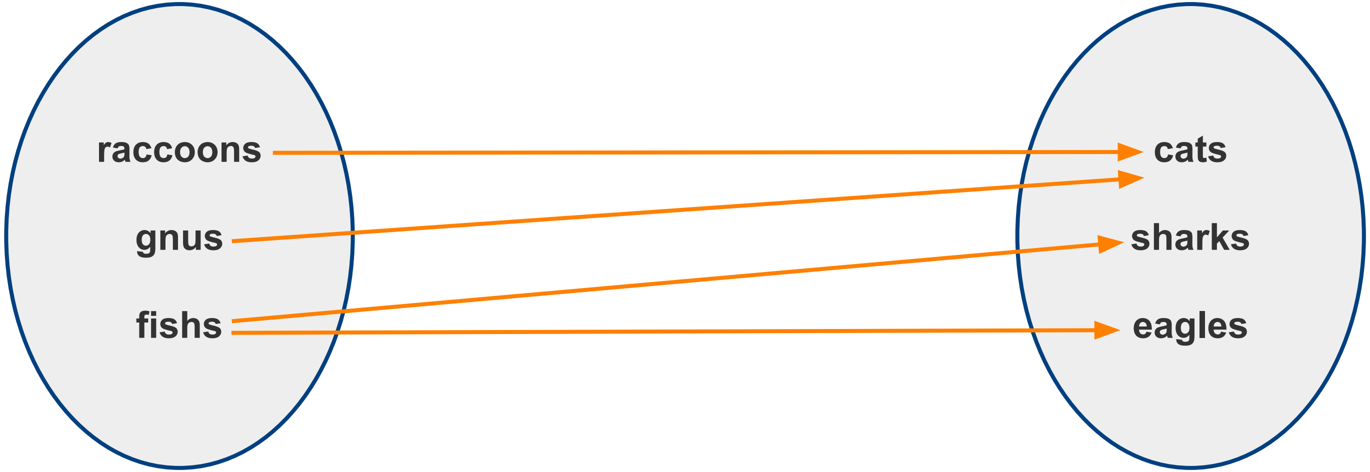
## Solve(transformation, output)

```
IN : transformation, output
OUT: input

dependencies = findDependencies(transformation)
for(dependency in dependencies)
    singleSolve(dependency, transformation)
```

# Dependency Discovery

# Many-to-Many Relationships



# Dependency Discovery

```
INPUT: transformation, input
OUTPUT: dependencies

output = function(input)
dependencies = {}

for each variable in input
     modifiedInput  = modify variable in input
     affectedOutput = differing variables between output and
                              transformation(modifiedInput)
     dependencies[variable] = affectedOutput
end
```

# Example Output

```
transformation = function(input) {
     var output = {};

     output.raccoons = input.raccoons. + “p”;
     output.cats     = 2 * input.cats;
     output.okapis   = input.cats + input.okapis;

     return output;
};
```

```
dependencies = {
     “raccoons”: [ “raccoons” ],
     “cats”: [ “cats” , “okapis” ],
     “okapis”: [ “okapis” ]
}
```

**needs to be transposed**

# Splitting Problem

# Problem

$f(x_0, x_1, \dots, x_m) = (y_0, y_1, \dots, y_n)$ → pretty **hard to solve**

Are there **easier subproblems**? Of course!

$f_0(x_0, x_1, \dots, x_m) = y_0$

$f_1(x_0, x_1, \dots, x_m) = y_1$

…

$f_n(x_0, x_1, \dots, x_m) = y_n$

# Simple(r) Problems

# Number-to-Number Relationships

**Easy**:

$f(x) = y \quad \text{with } x, y \in \mathbb{Z}$

Approach:

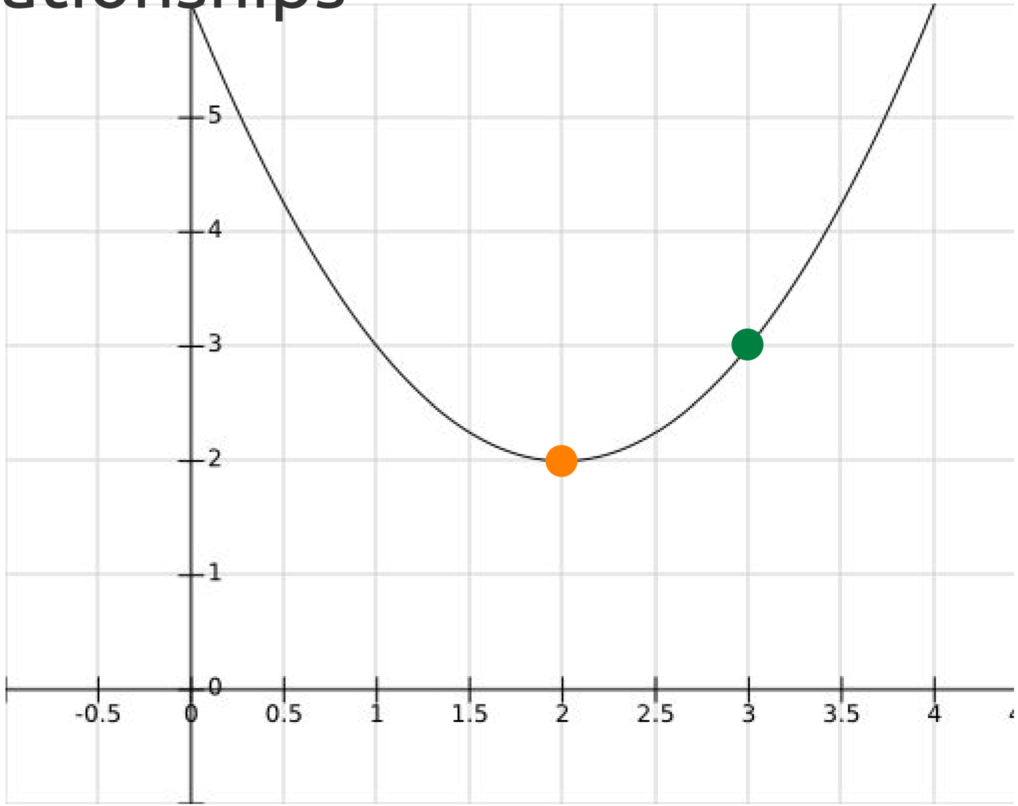
1. **Mutate** the input variable **systematically**.
2. **Observe change** in output. Does it get better?
3. **Repeat** until solution reached.

# Number-to-Number Relationships

Want to know x value of **green point** but know only y value.

Start with moving **orange point** in positive direction. (x+d)

Whenever the **orange point** passes the **green one**, swap direction and reduce step size.

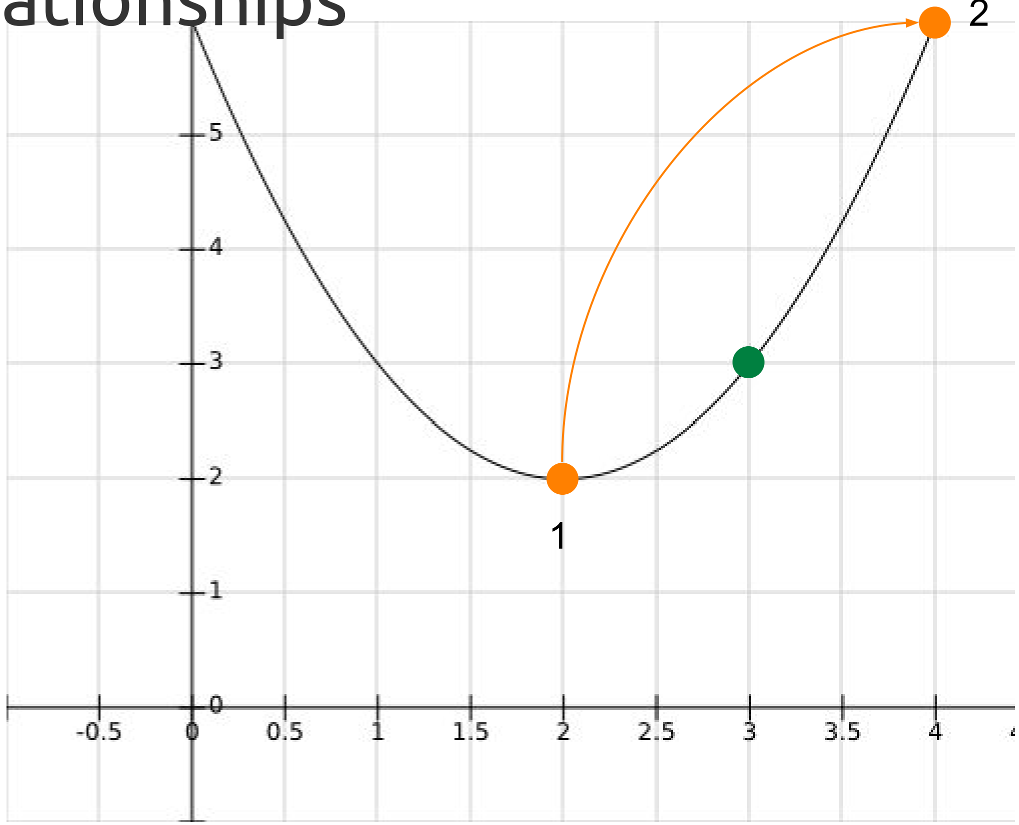


# Number-to-Number Relationships

Want to know x value of **green point** but know only y value.

Start with moving **orange point** in positive direction. (x+d)

Whenever the **orange point** passes the **green one**, swap direction and reduce step size.

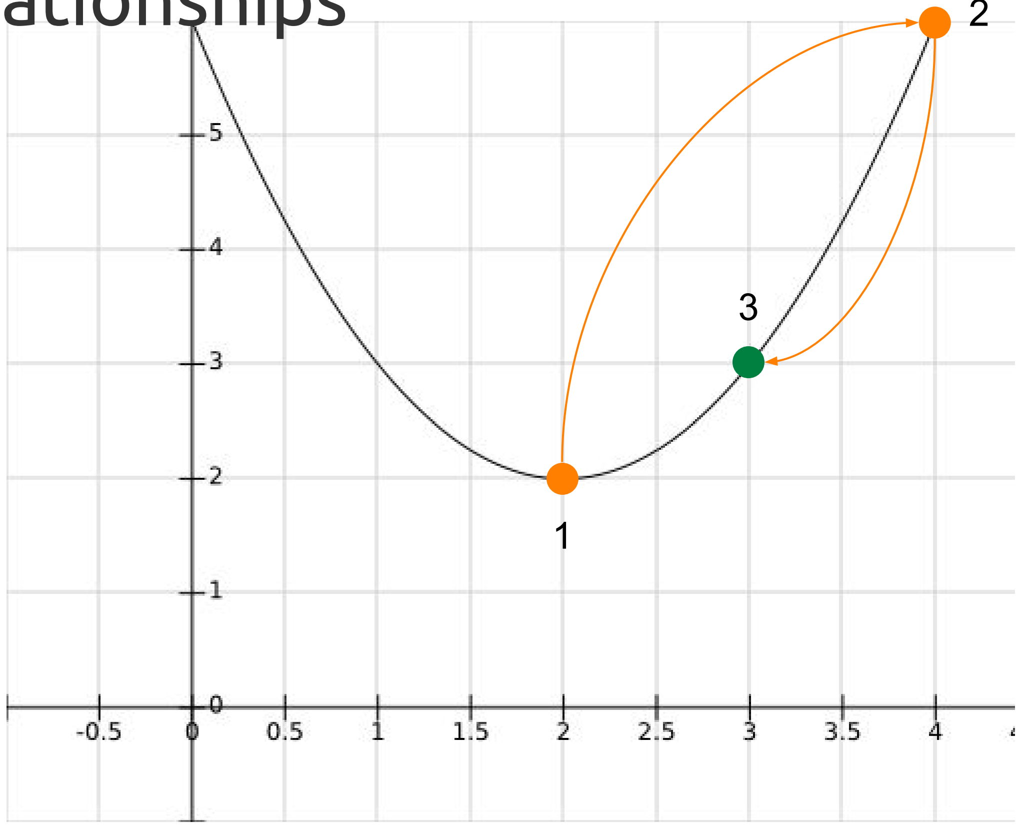


# Number-to-Number Relationships

Want to know x value of **green point** but know only y value.

Start with moving **orange point** in positive direction. (x+d)

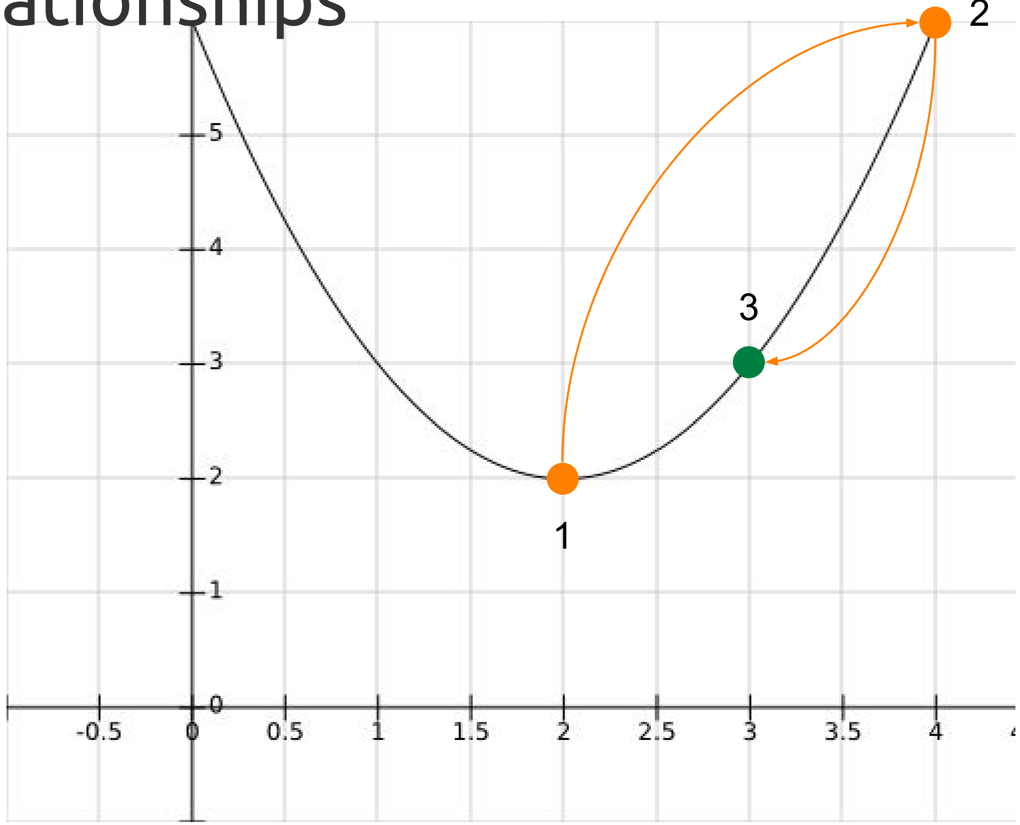
Whenever the **orange point** passes the **green one**, swap direction and reduce step size.



# Number-to-Number Relationships

Works for floats as well.

But what is with **strings**?



# Evolutionary Algorithms!

Inspired by Genetic.js: http://subprotocol.com/system/genetic-hello-world.html

But we changed the **fitness function**.

Also, we **do not know the string length** a priori.

→ Need to **change mutation function** as well to adjust string length.

# Fitness Function

| | | | | | |
|---|---|---|---|---|---|
| Solution: | H | A | L | L | O |
| Max. Fitness: | 127 | 127 | 127 | 127 | 127 |

= 5 * 127

= 635

# Fitness Function: Wrong Characters

| | | | | | |
|---|---|---|---|---|---|
| Solution: | H | A | L | L | O |
| Entity: | H | E | L | L | P |
| Fitness: | 127 | 123 | 127 | 127 | 126 |

= 630 / 635

# Fitness Function: Too Short

| | | | | | |
|---|---|---|---|---|---|
| Solution: | H | A | L | L | O |
| Entity: | H | E | L | L | |
| Fitness: | 127 | 123 | 127 | 127 | 0 |

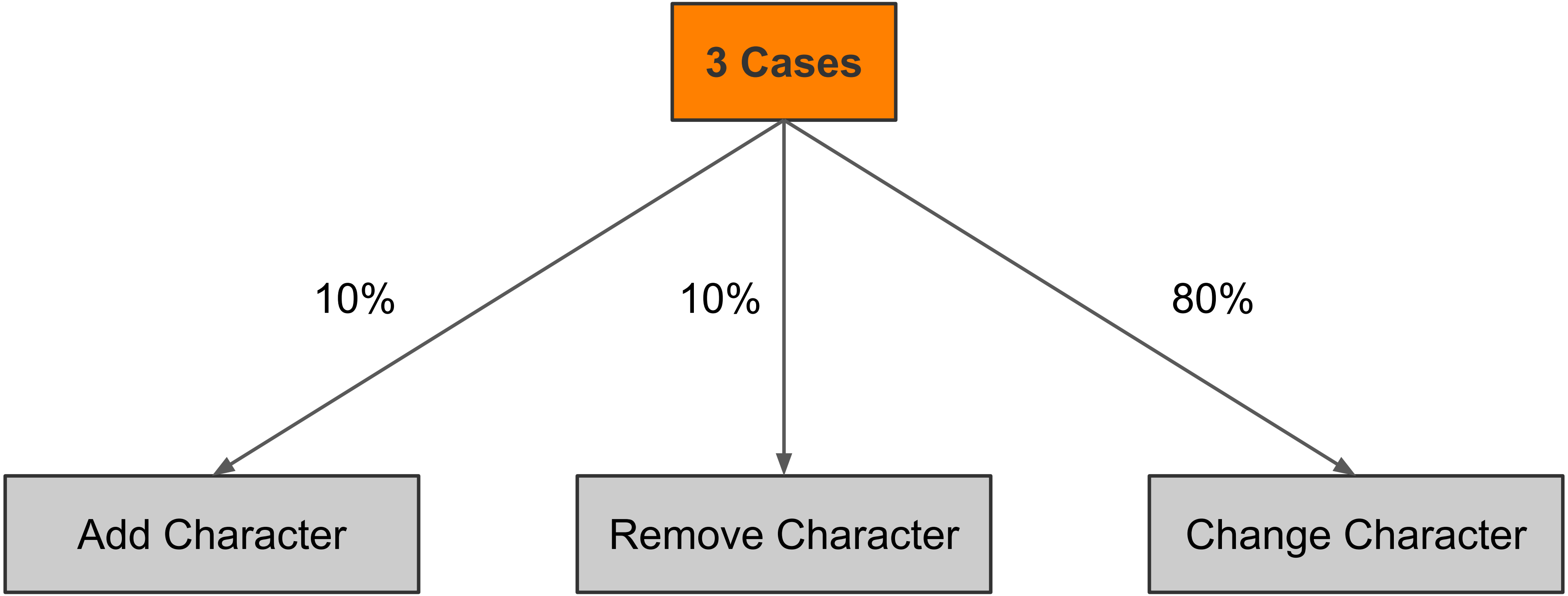
= 504 / 635

# Fitness Function: Too Long

| | | | | | | |
|---|---|---|---|---|---|---|
| Solution: | H | A | L | L | O | |
| Entity: | H | E | L | L | P | P |
| Fitness: | 127 | 123 | 127 | 127 | 126 | -127 |

= 503 / 635

# Mutation Function



# More Evolutionary Algorithms!

Now we can **solve string-to-string** functions.

Extending to **string-to-int** is easy.

→ Just change the fitness function!

# Fitness Function

```
var optimal = output;
var actual  = transformation(entity);

var fitness = Math.abs(optimal - actual);
```

# Many-to-One Relationships

**Complicated!** Yet easy:

```
var fitness = function(input){
     var optimal = output;
     var actual  = transformation(input);
     return Math.abs(actual - optimal);
};


var solution = numeric.uncmin(fitness, input).solution;
```

[2]

# Summary

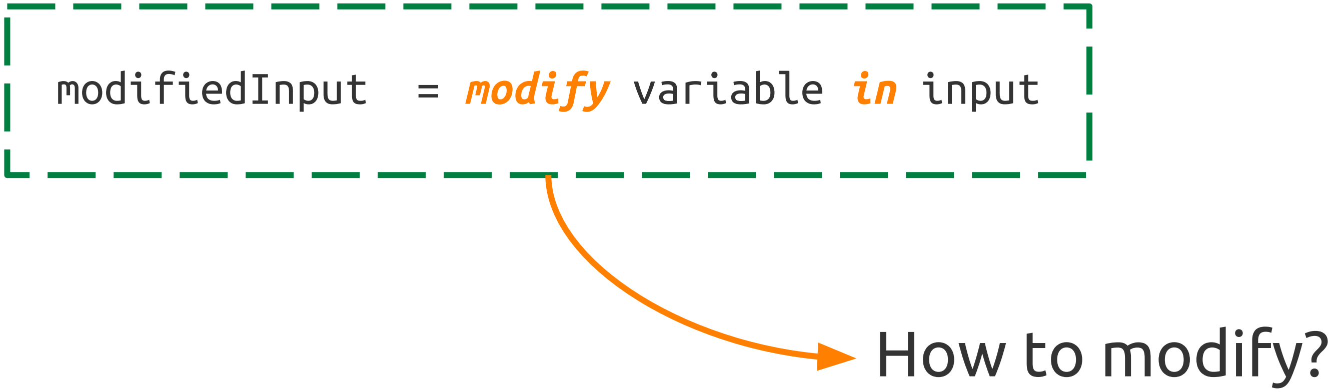
What are we doing?

1. find dependencies
2. apply **specific functions**

(Some dependencies may be insoluble.)

# Future Work

# Dependency Discovery



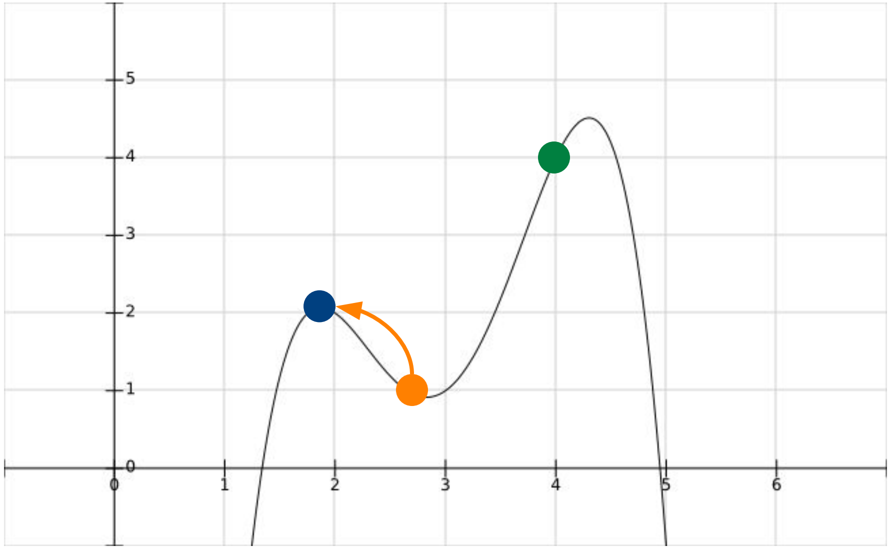
```
modifiedInput  = modify variable in input
```

How to modify?

# Going Crazy

Evolutionary algorithms tend to find **local optima** instead of global one.

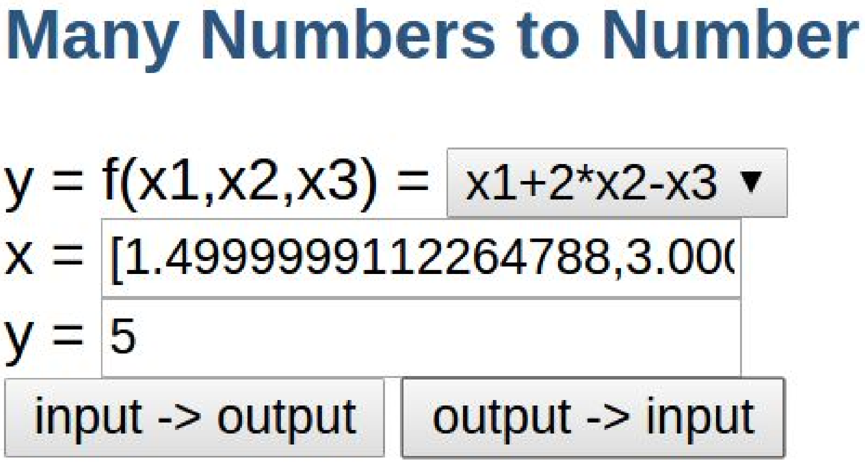
Infrequent, **heavy mutation** might solve this issue.



# Rectifying Floats

**uncmin** produces floating-point artifacts

We have to **rectify** them in a post-processing step.



## Many Numbers to Number

y = f(x1,x2,x3) = x1+2*x2-x3 ▼

x = [1.4999999112264788,3.000

y = 5

# Sources

[1] https://alpha.trycarbide.com/
retrieved: 2017-01-09

[2] http://www.numericjs.com/documentation.html
retrieved: 2017-01-11