On Fitness Functions...

In order for us to implement a genetic metaphor, we need a way to evaluate the **fitness** of each member of the population at periodic intervals.

Here is one possible definition for a **fitness function** a robot solving the "boundary following" problem:

> the number of cells next to walls that a rule visits during ten iterations of sixty moves each, with each iteration starting at a random position

Exercise: Finding Fitness Functions

Work in groups of three or four to...

Propose a fitness function each of these problems:

- controlling an elevator in an n-story elevator
- controlling the stoplights at a four-way intersection

Attack the problems in this order:

- Determine what data you would collect about each program execution.

  That is, what matters?

- Determine how you would combine the data into a single answer.

  That is, how much does each matter?

Toward a Solution: Elevators

Factors that indicate the quality of elevator service:

- average waiting time before the elevator arrives (on)
- average waiting time before it arrives (off)
- maximal waiting times

During the early stages of evolving an elevator controller, some of these may be infinite because some passengers may never be serviced.

So, we might also include factors such as number of passengers that are never serviced.

Toward a Solution: Stoplights

Factors that indicate the quality of stoplight service:

- average waiting times at stoplights for through traffic
- maximal waiting times at stoplights for through traffic
- average waiting times for side-street traffic
- maximal waiting times for side-street traffic

Sidestreet and through traffic are in conflict, so the fitness function must take into account the preferred trade-off.

We might also consider the volume of traffic along the main street under different traffic conditions (heavy, medium, slow).

How good are your fitness functions?
How do you know?

A Variation: Genetic Programming

Rules are like programs; why not apply the idea of genetic evolution directly to programs?

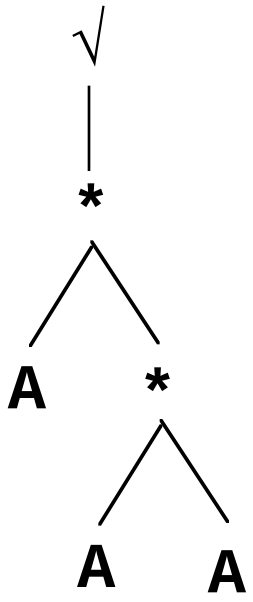
- broaden the alphabet to functions and terminals, where terminals are constants and variables
- broaden from one "link" to many

As a simple example, consider a program to compute the orbital period P of a planet, given the planet's average distance from the sun A.

If we express P and A in units relative to Earth's values, then the "right answer" is:

$$P = \sqrt{A^3}$$

Suppose that we wished to learn this formula, based on empirical data gathered from sky gazing?



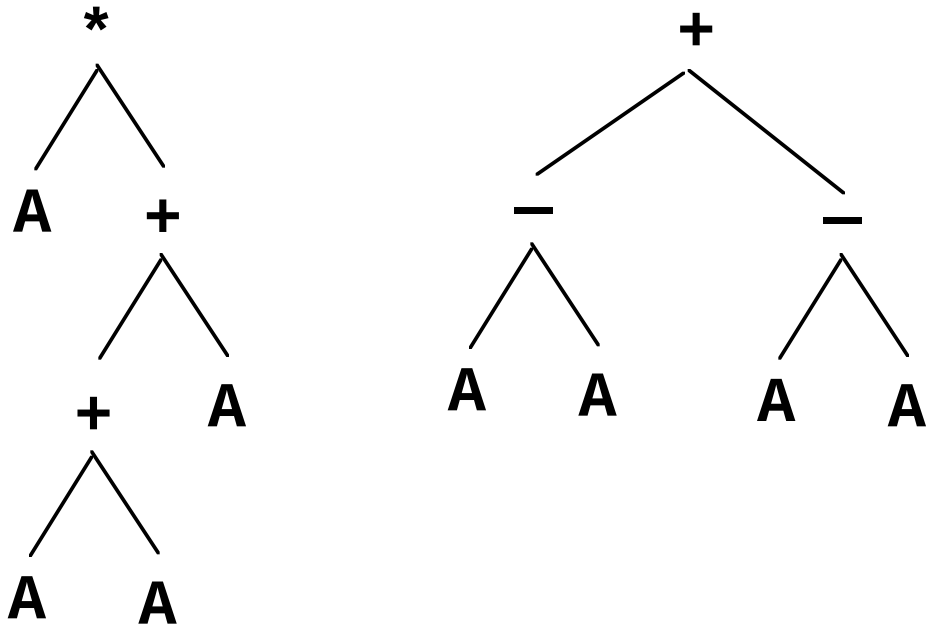
Setting Up The Learning Environment

1. *Identify the set of functions and terminals that you will allow in the program.*

{ +, -, *, /, sqrt, ...} **{A, ...}**

2. *Generate an initial population of (perhaps random) programs created out of the alphabet.*

Here are some examples I generated randomly:



Computing the Fitness of Programs

3. Calculate the fitness of each program in the population by running it on a set of training cases.

PLANET	A	Actual P
Venus	0.72	0.61
Earth	1.00	1.00
Mars	1.52	1.84
Jupiter	5.20	11.90
Saturn	9.53	29.40
Uranus	19.10	83.50

Fitness function: — total error

PLANET	A	Actual P	PROGRAM 1 ERROR	PROGRAM 2 ERROR
Venus	0.72	0.61	0.9452	0.61
Earth	1.00	1.00	2.0000	1.00
Mars	1.52	1.84	5.0912	1.84
Jupiter	5.20	11.90	69.2200	11.90
Saturn	9.53	29.40	243.0627	29.40
Uranus	19.10	83.50	1010.9300	83.50
			---------	---------
			1331.2491	128.25

More on Computing Fitness

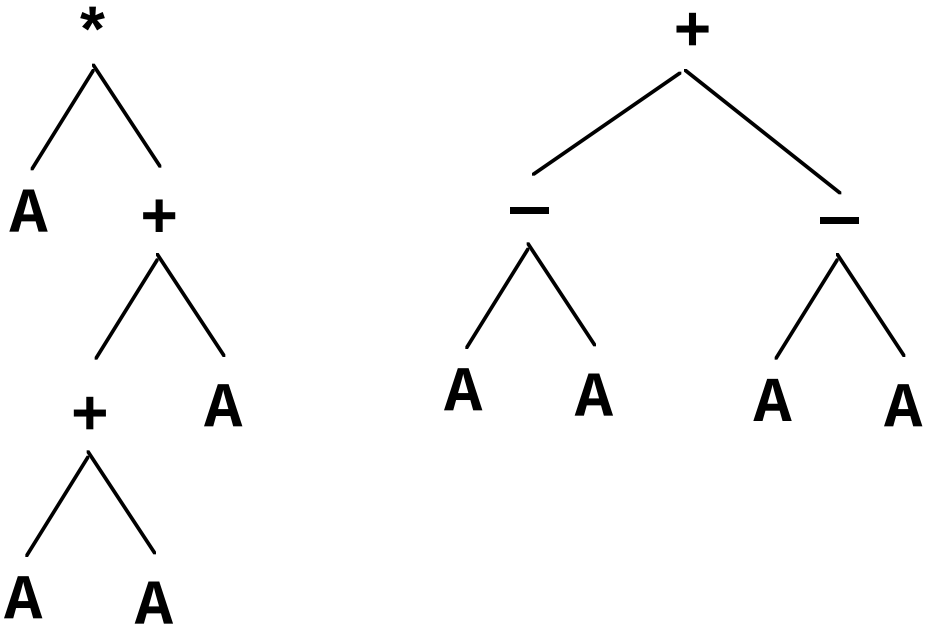
PLANET	A	Actual P
Venus	0.72	0.61
Earth	1.00	1.00
Mars	1.52	1.84
Jupiter	5.20	11.90
Saturn	9.53	29.40
Uranus	19.10	83.50

Fitness function:
Percentage of test cases with 10% of correct answer

PLANET	A	Actual P	PROGRAM 1 ERROR	PROGRAM 2 ERROR
Venus	0.72	0.61	0.9452	0.61
Earth	1.00	1.00	2.0000	1.00
Mars	1.52	1.84	5.0912	1.84
Jupiter	5.20	11.90	69.2200	11.90
Saturn	9.53	29.40	243.0627	29.40
Uranus	19.10	83.50	1010.9300	83.50

Generating the Next Generation

4. *Apply genetic operators to the population to create a new population. Use a program's fitness as its strength when choosing programs to reproduce and replace.*



If you generate an initial population of sufficient size and diversity, then you can build a system with no mutation and no direct reproduction.

All reproduction is by crossover.