15: Genetic Programming 1

- Motivation
- Machine learning
- Representation
- Textbook chapter 6.4
- Reference book chapter 5
- Reference book: *Genetic Programming - An Introduction, Banzhaf et al, Morgan Kaufmann*

Prof. Ting Hu, School of Computing, Queen’s University

Motivation

- Software consistently lags years behind the capabilities of the hardware
- The “craftsman” approach for computer programming
- Development of computer programming: structured programming, object-oriented programming, object libraries, rapid prototyping, visualized programming, AI-assisted programming…

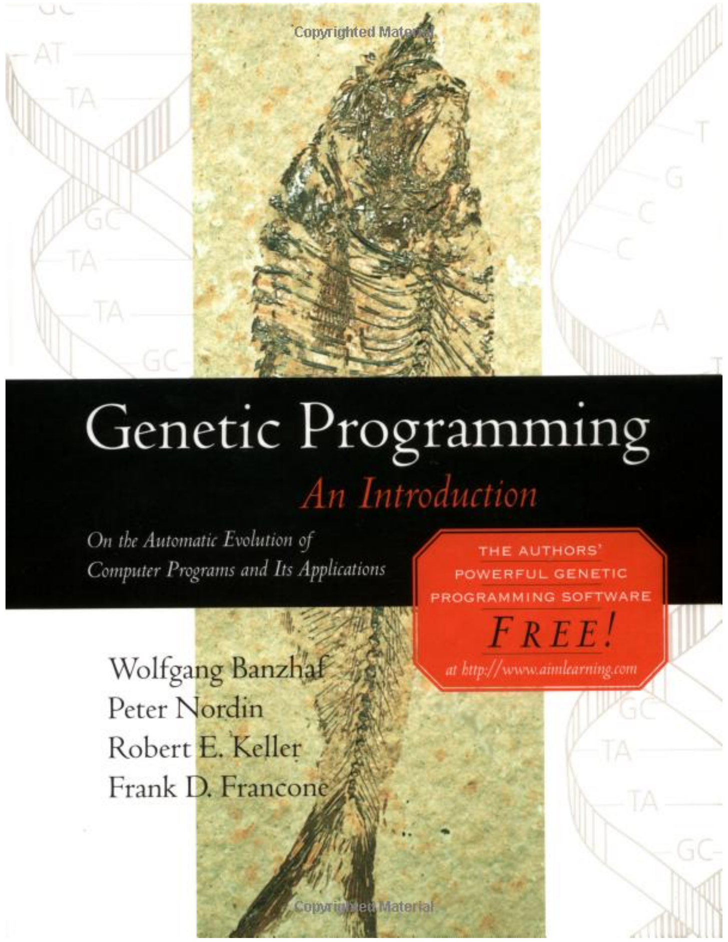
Machine Learning definition

- Arthur Samuel (1959): Field of study that gives computers the ability to learn without being explicitly programmed.

- Tom Mitchell (1996): The study of computer algorithms that improve automatically through experience.

Genetic programming

GP: to induce a population of computer programs or programming language structures that improve automatically as they experience the data on which they are trained



Machine Learning

- Training, testing, and cross validation
- Supervised learning
    - each training instance is an input accompanied by the correct output
- Unsupervised learning
    - the ML system is not told what the correct output is and itself looks for patterns in the input data
- Reinforcement learning
    - a general signal (unspecified) for quality of an output is fed back to the learning algorithm

Machine Learning (cont.d)

- Classification
    - supervised, discrete output (usually binary)
- Regression
    - supervised, continuous output
- Clustering
    - unsupervised
- Dimensionality reduction
    - mapping high-dimensional inputs into a lower-dimensional space

GP quick overview

- Developed in the U.S.A. in the 1990's
- Early names: John Koza
- Typically applied to:
  - machine learning tasks (regression, classification…)
- Attributed features:
  - competes with neural nets and alike
  - needs huge populations (thousands)
  - slow
- Special:
  - non-linear chromosomes like trees and graphs

GP technical summary table

Representation	Tree structures
Recombination	Exchange of subtrees
Mutation	Random change in trees
Parent selection	Fitness proportional
Survivor selection	Generational replacement

Introductory example: credit scoring

- Bank wants to distinguish good from bad loan applications
- Model needed that matches historical data

ID	No of children	Salary	Marital status	OK?
ID-1	2	45000	Married	0
ID-2	0	30000	Single	1
ID-3	1	40000	Divorced	1
…				

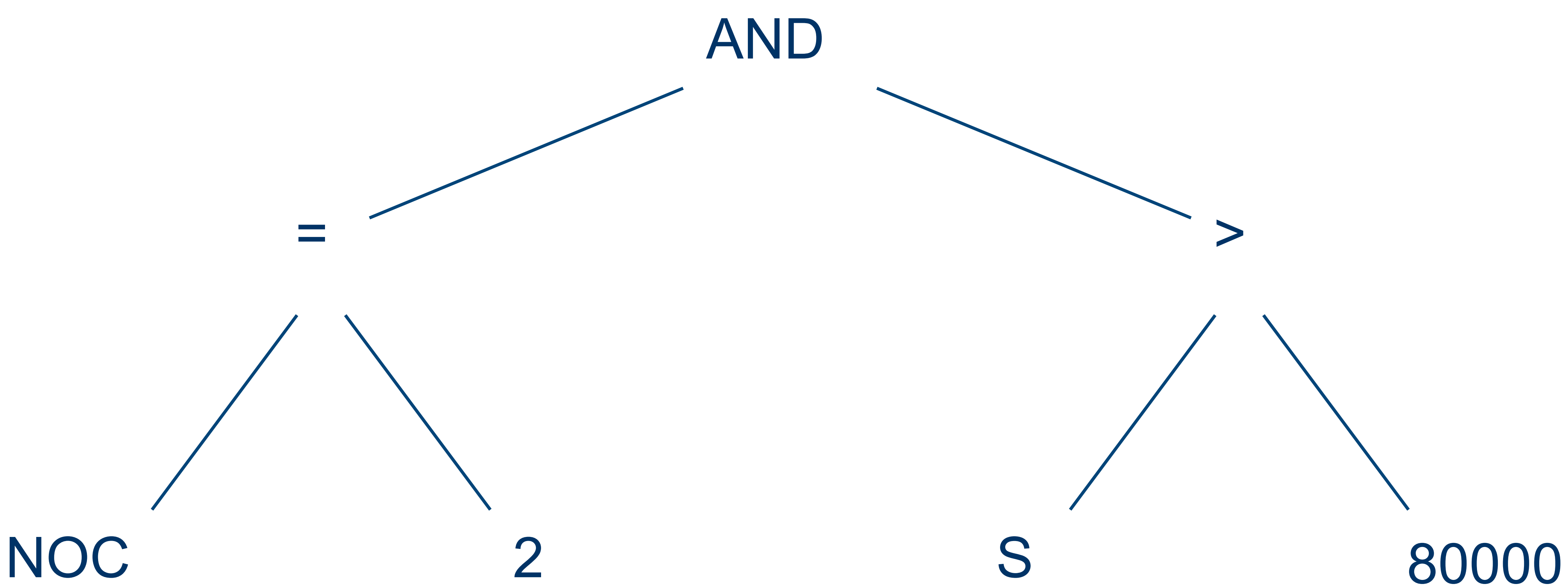
Introductory example: credit scoring

- A possible model:
  - IF (NOC = 2) AND (S > 80,000) THEN good ELSE bad
- In general:
  - IF formula THEN good ELSE bad
- Only unknown is the right formula, hence
  - Our search space (phenotypes) is the set of formulas
- Natural fitness of a formula: percentage of well classified cases of the model it stands for
- Natural representation of formulas (genotypes) is: parse trees

Introductory example: credit scoring

IF (NOC = 2) AND (S > 80,000) THEN good ELSE bad

can be represented by the following tree



Tree-based GP representation

- Trees are a universal form, e.g. consider
  - Arithmetic formula

$$2 \cdot \pi + \left((x + 3) - \frac{y}{5 + 1} \right)$$

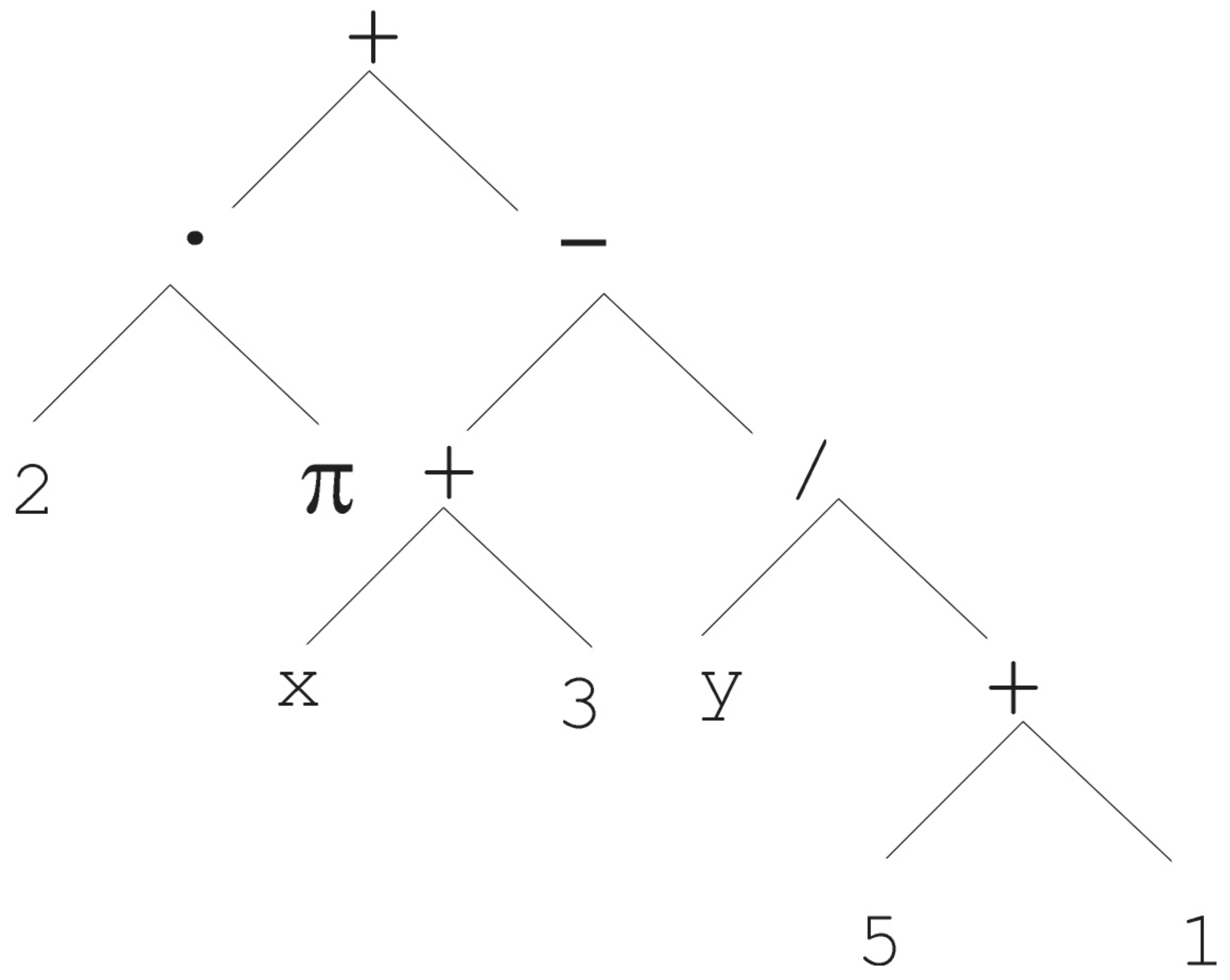
  - Logical formula

$$(x \wedge \text{true}) \rightarrow ((x \vee y) \vee (z \leftrightarrow (x \wedge y)))$$

  - Program

```
i =1;
while (i < 20)
{
        i = i +1
}
```

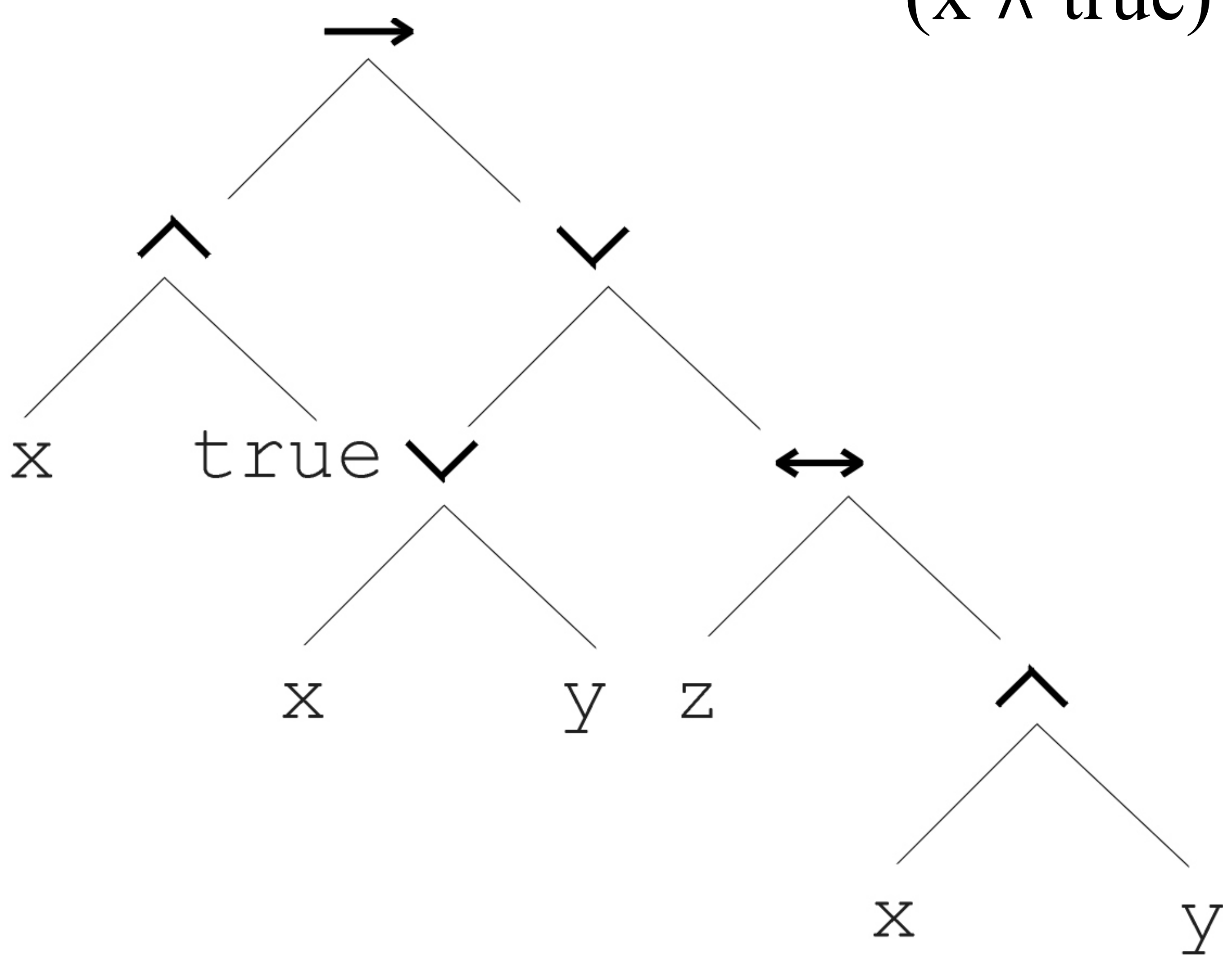
Arithmetic



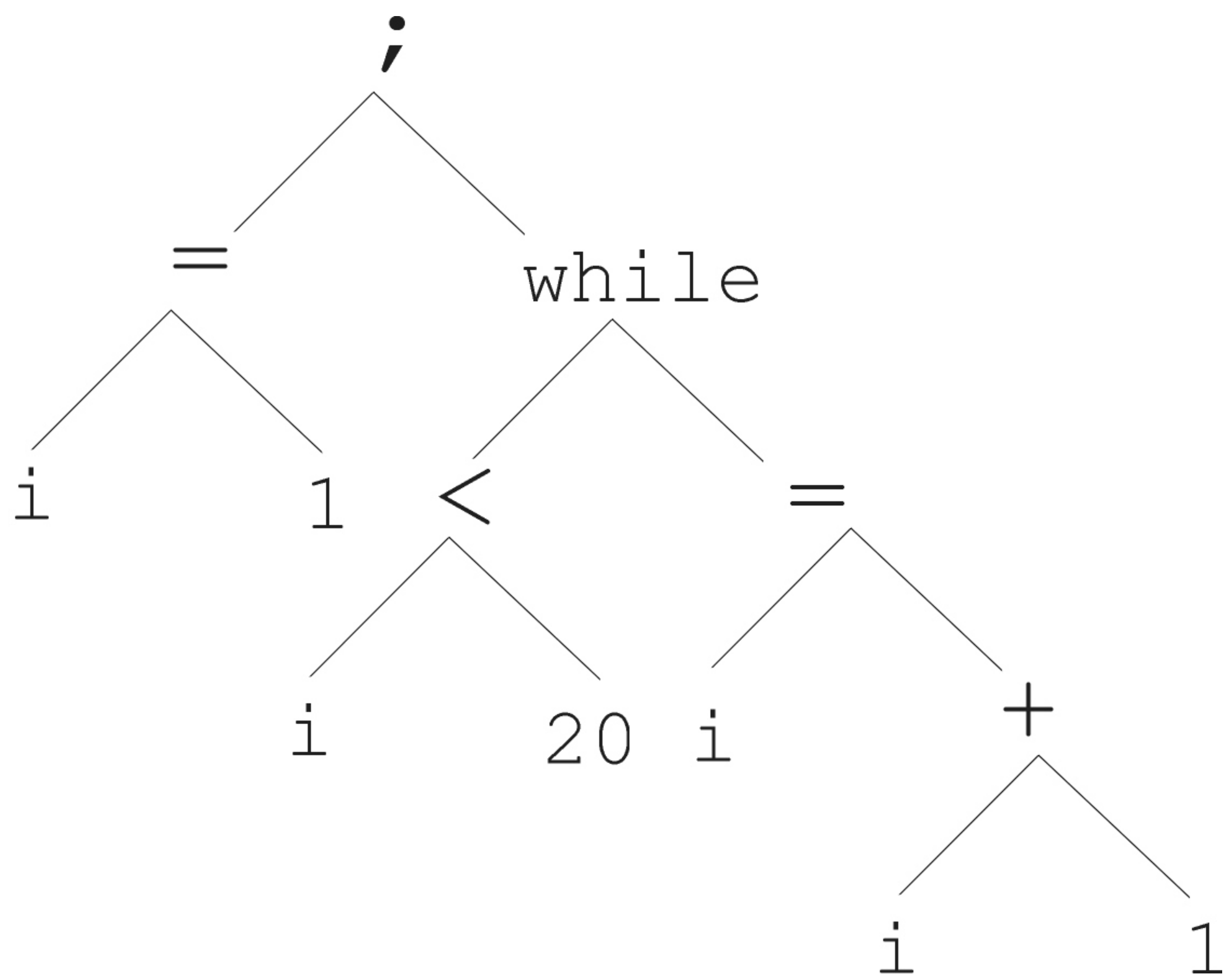
$$2 \cdot \pi + \left((x+3) - \frac{y}{5+1} \right)$$

Logic

$$(x \wedge \text{true}) \rightarrow ((x \vee y) \vee (z \leftrightarrow (x \wedge y)))$$



Programs




```
i =1;
while (i < 20)
{
        i = i +1
}
```

GP representation

- In GA and ES, chromosomes are linear structures (binary strings, integer strings, real-valued vectors, permutations)
- Tree shaped chromosomes are non-linear structures
- In GA and ES, the size of the chromosomes is fixed
- Trees in GP may vary in depth and width

GP terminals

- Symbolic expressions can be defined by
  - Terminal set T
  - Function set F (with the arities of function symbols)
- Terminals provide a value to the system
  - comprised of the inputs to the GP program, the constants, and the zero-argument functions
  - leave nodes of a tree, i.e., terminate a branch
  - features of the learning domain

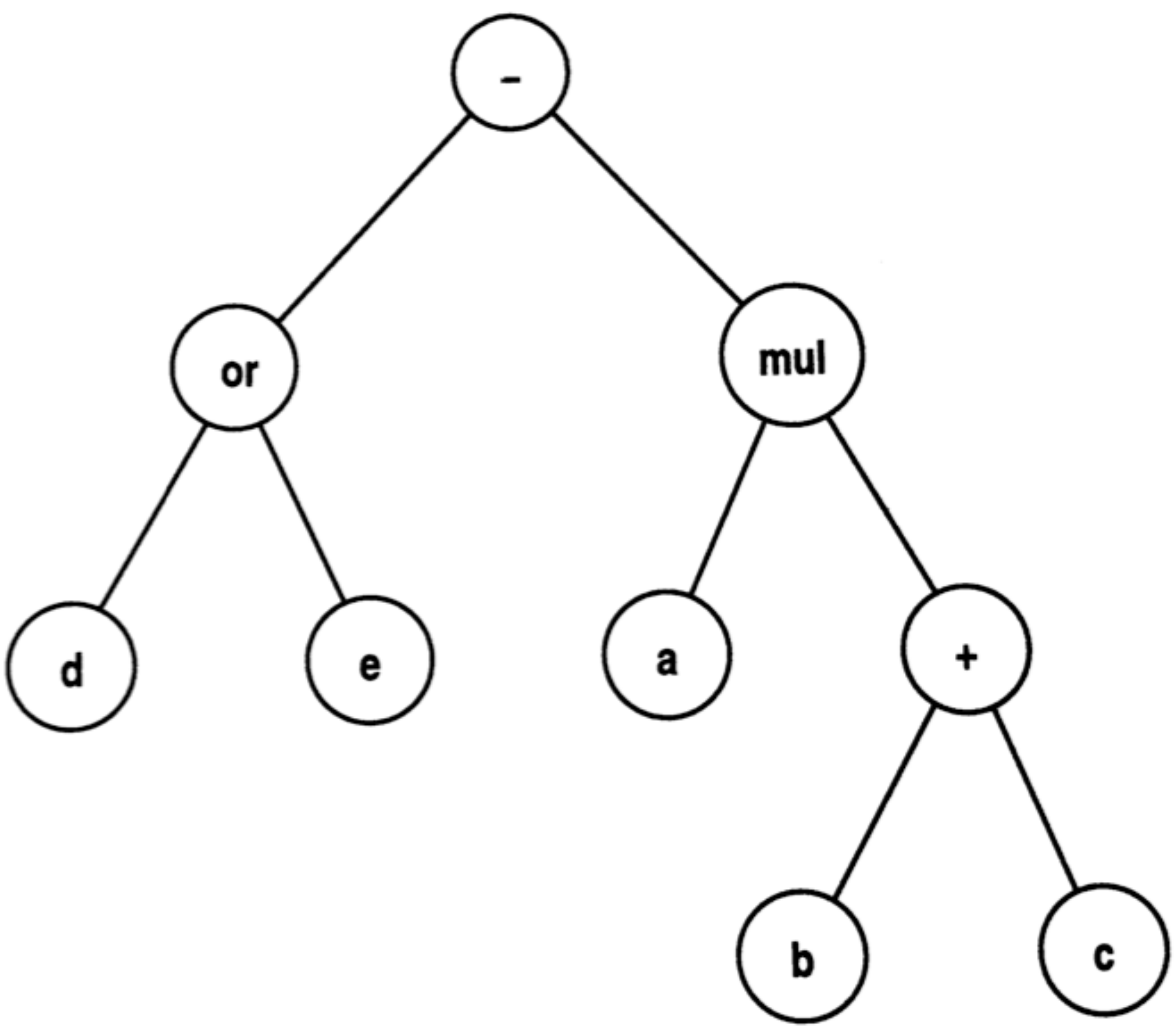
GP functions

- Functions process a value already in the system
  - comprised of the statements, operators, and functions available to the GP system
  - application-specific, selected to fit the problem domain
  - Boolean, arithmetic, transcendental functions
  - assignment, conditional, loop statements
  - subroutines, e.g., read sensor, turn left, move ahead in robotics

Choosing terminals and functions

- Sufficiency and parsimony
  - for choosing both functions and constants
- Adopting the following general recursive definition:
  - every $t \in T$ is a correct expression
  - $f(e_1, \ldots, e_n)$ is a correct expression if $f \in F$, arity(F) = n and $e_1, \ldots e_n$ are correct expressions
  - There are no other forms of correct expressions
- *Closure* property: any function should be able to handle all values it might receive as inputs
  - e.g. division operator -> protected division

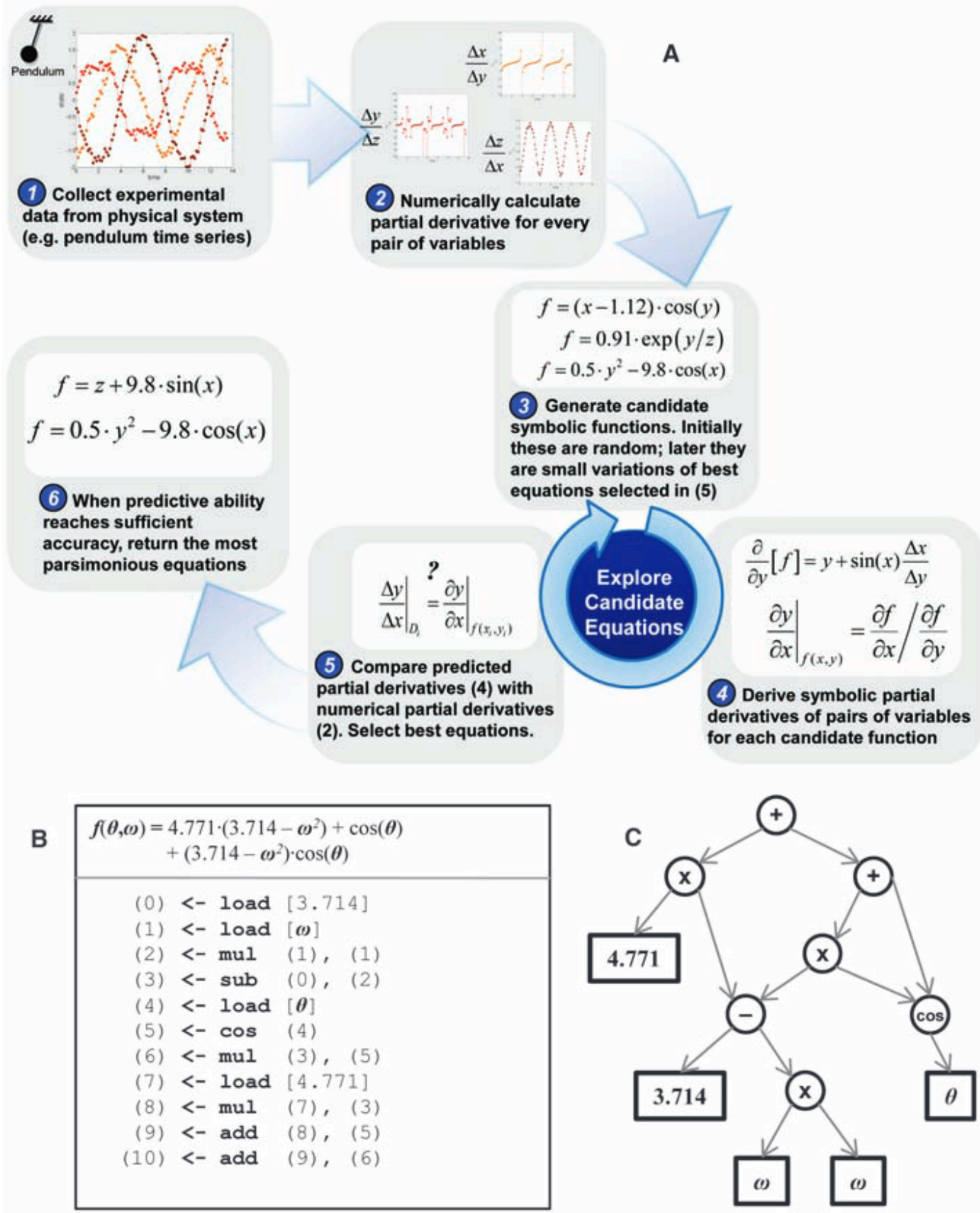
Execution of GP trees



Distilling Free-Form Natural Laws from Experimental Data

Michael Schmidt[1] and Hod Lipson[2,3]*

For centuries, scientists have attempted to identify and document analytical laws that underlie physical phenomena in nature. Despite the prevalence of computing power, the process of finding natural laws and their corresponding equations has resisted automation. A key challenge to finding analytic relations automatically is defining algorithmically what makes a correlation in observed data important and insightful. We propose a principle for the identification of nontriviality. We demonstrated this approach by automatically searching motion-tracking data captured from various physical systems, ranging from simple harmonic oscillators to chaotic double-pendula. Without any prior knowledge about physics, kinematics, or geometry, the algorithm discovered Hamiltonians, Lagrangians, and other laws of geometric and momentum conservation. The discovery rate accelerated as laws found for simpler systems were used to bootstrap explanations for more complex systems, gradually uncovering the "alphabet" used to describe those systems.


A
Pendulum
1 Collect experimental data from physical system (e.g. pendulum time series)
2 Numerically calculate partial derivative for every pair of variables
$\frac{\Delta x}{\Delta y}$
$\frac{\Delta y}{\Delta z}$
$\frac{\Delta z}{\Delta x}$
$f=(x-1.12)\cdot\cos(y)$
$f=0.91\cdot\exp(y/z)$
$f=0.5\cdot y^2-9.8\cdot\cos(x)$
3 Generate candidate symbolic functions. Initially these are random; later they are small variations of best equations selected in (5)
Explore Candidate Equations
$\frac{\partial}{\partial y}[f]=y+\sin(x)\frac{\Delta x}{\Delta y}$
$\left.\frac{\partial y}{\partial x}\right|_{f(x,y)}=\frac{\partial f}{\partial x}\Big/\frac{\partial f}{\partial y}$
4 Derive symbolic partial derivatives of pairs of variables for each candidate function
$\left.\frac{\Delta y}{\Delta x}\right|_{D_i}\overset{?}{=}\left.\frac{\partial y}{\partial x}\right|_{f(x_i,y_i)}$
5 Compare predicted partial derivatives (4) with numerical partial derivatives (2). Select best equations.
$f=z+9.8\cdot\sin(x)$
$f=0.5\cdot y^2-9.8\cdot\cos(x)$
6 When predictive ability reaches sufficient accuracy, return the most parsimonious equations
B
$f(\theta,\omega)=4.771\cdot(3.714-\omega^2)+\cos(\theta)+(3.714-\omega^2)\cdot\cos(\theta)$
(0) <- load [3.714]
(1) <- load [ω]
(2) <- mul (1), (1)
(3) <- sub (0), (2)
(4) <- load [θ]
(5) <- cos (4)
(6) <- mul (3), (5)
(7) <- load [4.771]
(8) <- mul (7), (3)
(9) <- add (8), (5)
(10) <- add (9), (6)
C
+
x
+
4.771
x
−
cos
3.714
x
θ
ω
ω