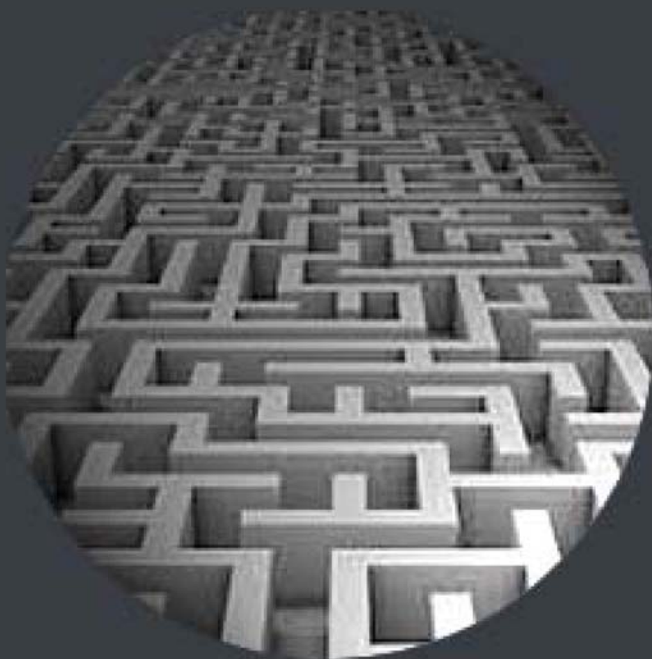
University of BRISTOL
Department of Computer Science

PROGRAMMING and ALGORITHMS II



Brief Recap on

COMPLEXITY BASICS

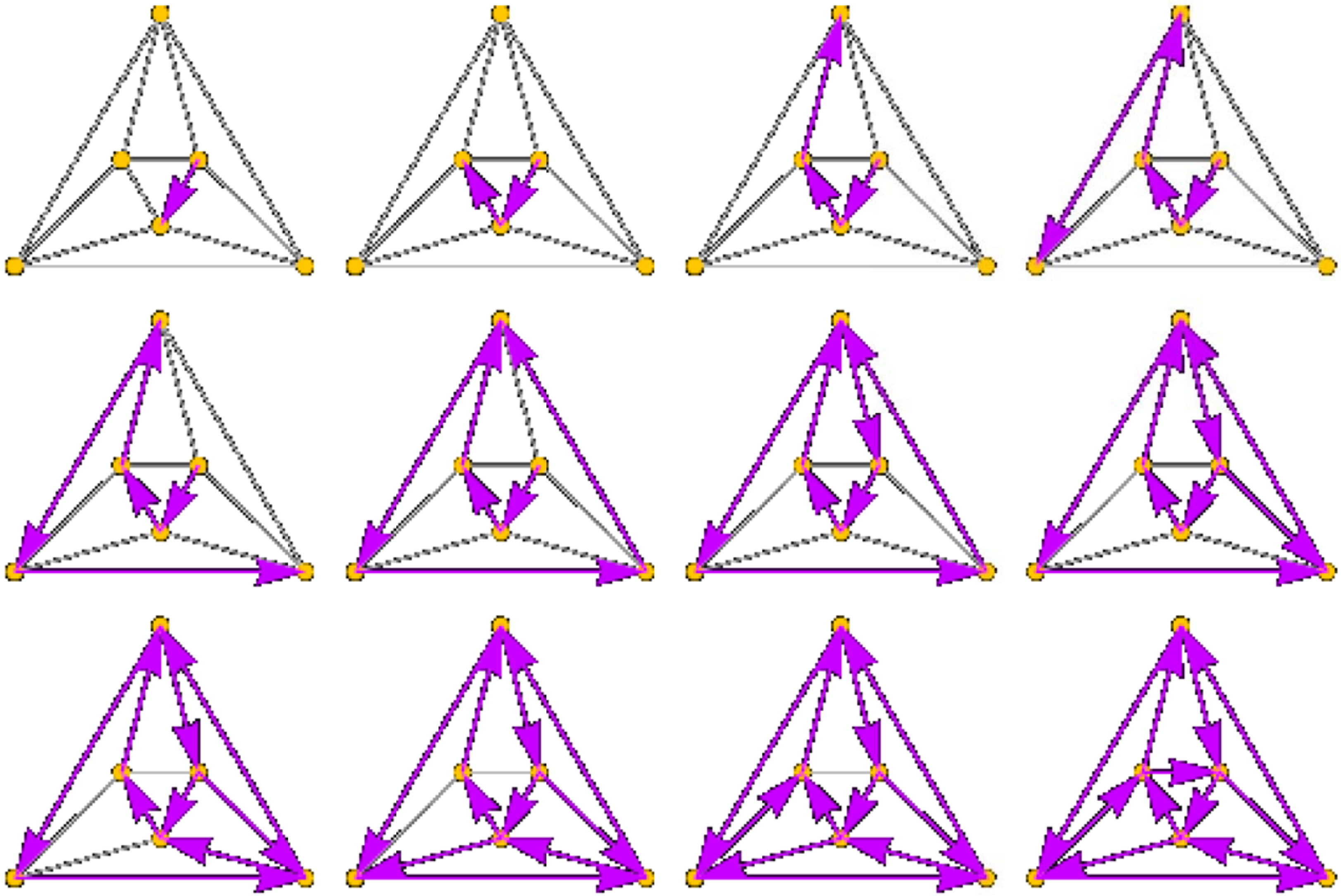
Dr Tilo Burghardt

Unit Code COMS10001

Eulerian Cycles

(use each edge once to cycle the graph)

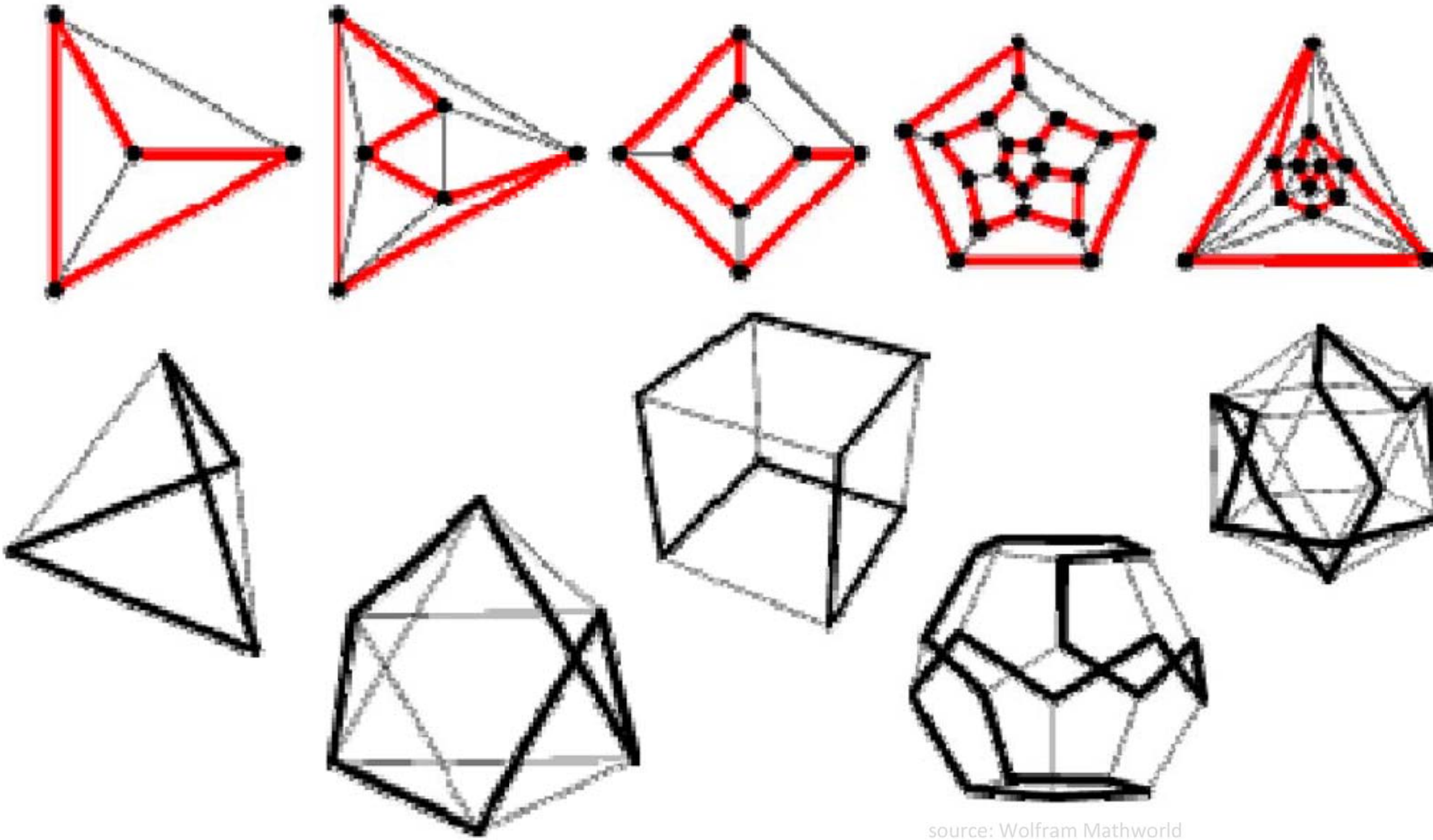
University of BRISTOL | Department of Computer Science



source: Wolfram Mathworld

Hamiltonian Cycles

(use each node once to cycle the graph)



source: Wolfram Mathworld

University of BRISTOL | Department of Computer Science

Computability Classes...

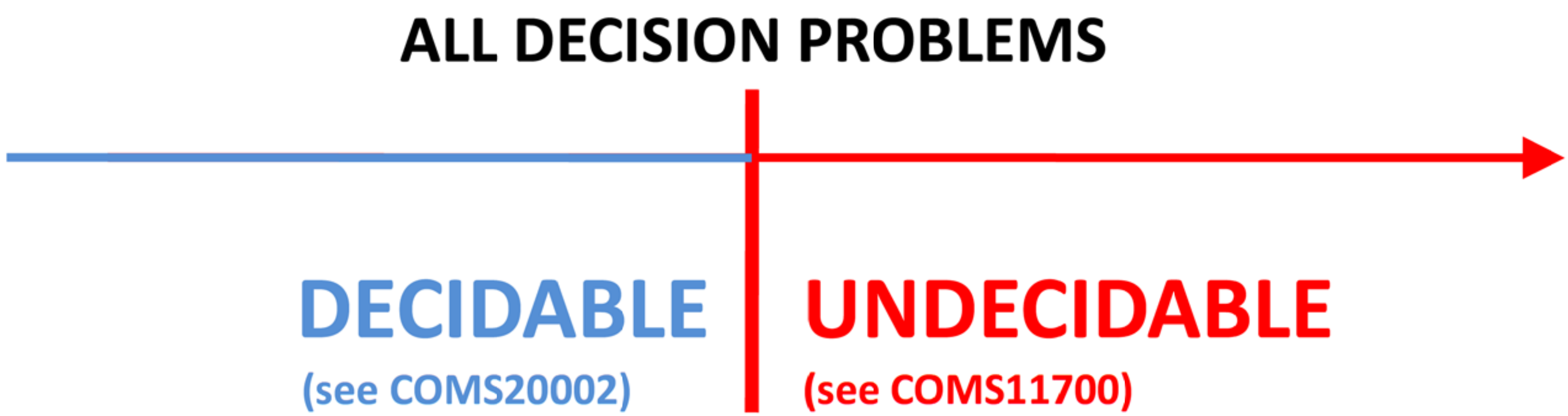
describe sets of problems (or languages) that can be solved (decided/recognised) **at all** by a given machine (e.g. a Turing Machine)...

Complexity Classes...

describe sets of problems (or languages) that can be solved by a given machine (e.g. a Turing Machine) in a **bounded amount** of time, space or other resource...

Computability Theory

- Computability Theory aims at working out what can be computed at all
- Not everything can be computed! – some problems/languages cannot be decided by a Turing Machine ... e.g. halting problem (board)



Recap: Chomsky Hierarchy

Undecidable

TM

- Type 0: recursively enumerable
  $X \rightarrow Y$

Decidable

- Type 1: context-sensitive
  $XAY \rightarrow XZY$

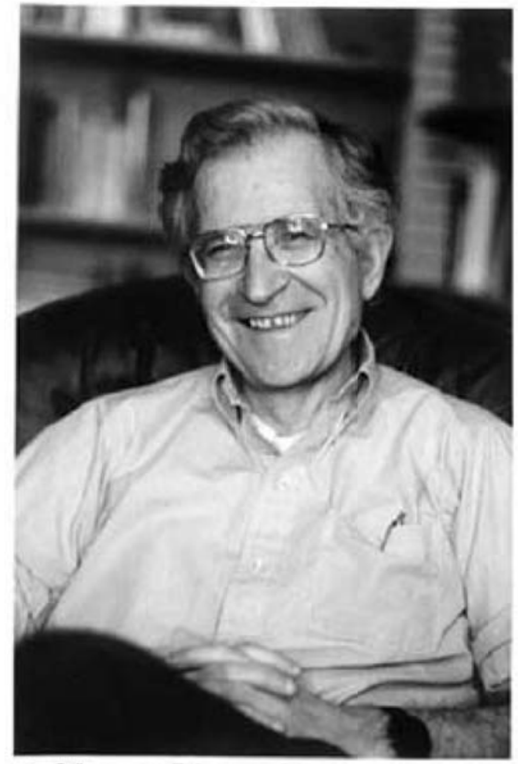
PDA

- Type 2: context-free
  $A \rightarrow X$

DFA

- Type 3: regular
  $A \rightarrow a\, ; A \rightarrow aB$

trivial



A Noam Chomsky

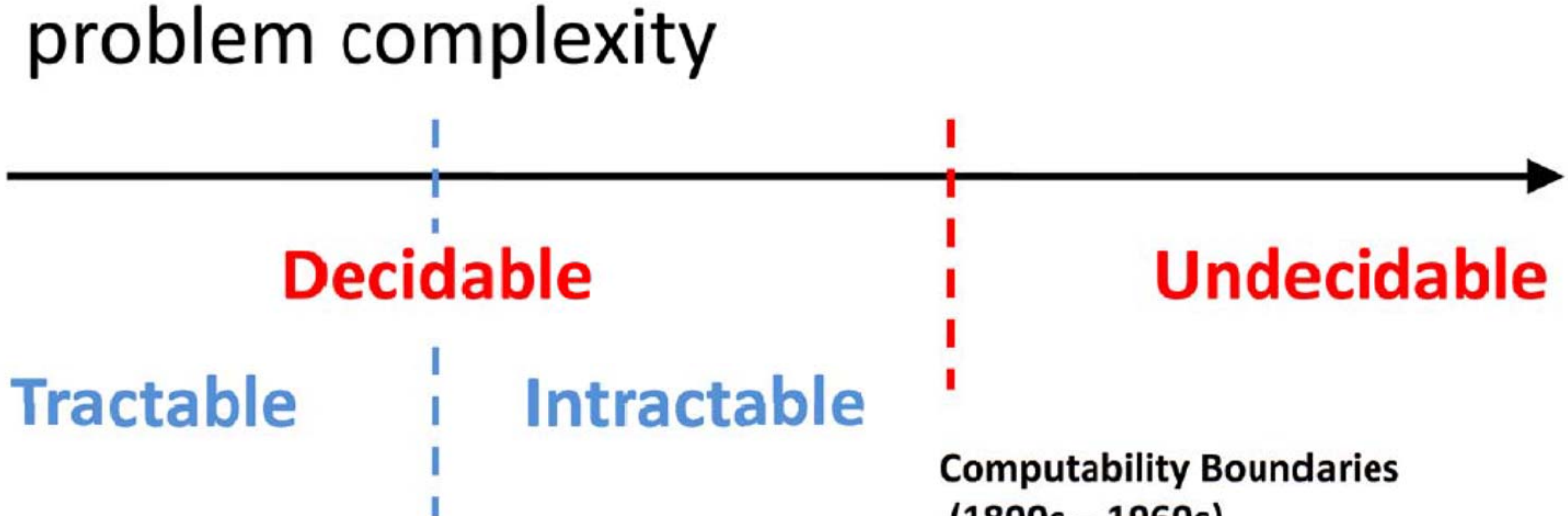
$X, Y, Z \ldots$ strings of terminals and non-terminals

a, b, c … terminals

$A, B, C \ldots$ non-terminals

Modelling Problem Difficulty

University of BRISTOL
Department of Computer Science



Complexity Boundaries (1960s – today)

1960s: Hartmanis and Stearns: Complexity classes
1971: Cook/Levin, Karp: $P=NP$?
1976: Knuth's O, Ω, Θ

Computability Boundaries (1800s – 1960s)

1900: Hilbert's Problems
1936: Turing's *Computable Numbers*
1957: Chomsky's *Syntactic Structures*

Recap: Big-O and Big-Ω

$f(n) = O(g(n))$ defines an upper bound and means:

There are *positive* constants c and n_0 such that

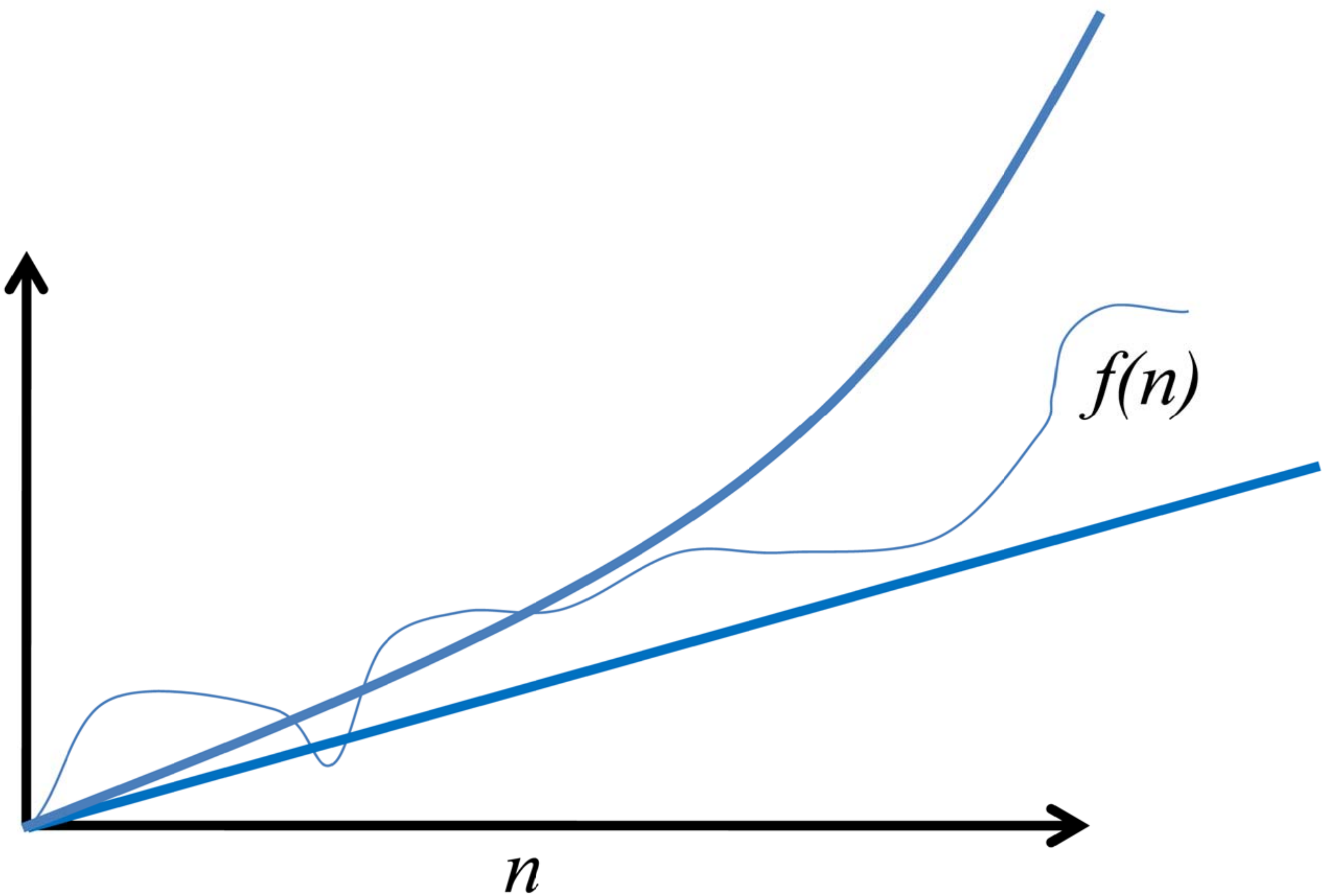
$$f(n) \leq cg(n) \quad \text{for all } n \geq n_0$$

$f(n) = \Omega(g(n))$ defines a lower bound and means:

There are *positive* constants c and n_0 such that

$$f(n) \geq cg(n) \quad \text{for all } n \geq n_0$$

Recap: Big-O and Big-Ω



Theta (`order of’) Notation

University of BRISTOL Department of Computer Science

Intuitively: $f(n) = \Theta(g(n))$ stipulates a set of functions that <u>grow as fast as</u> f that is:

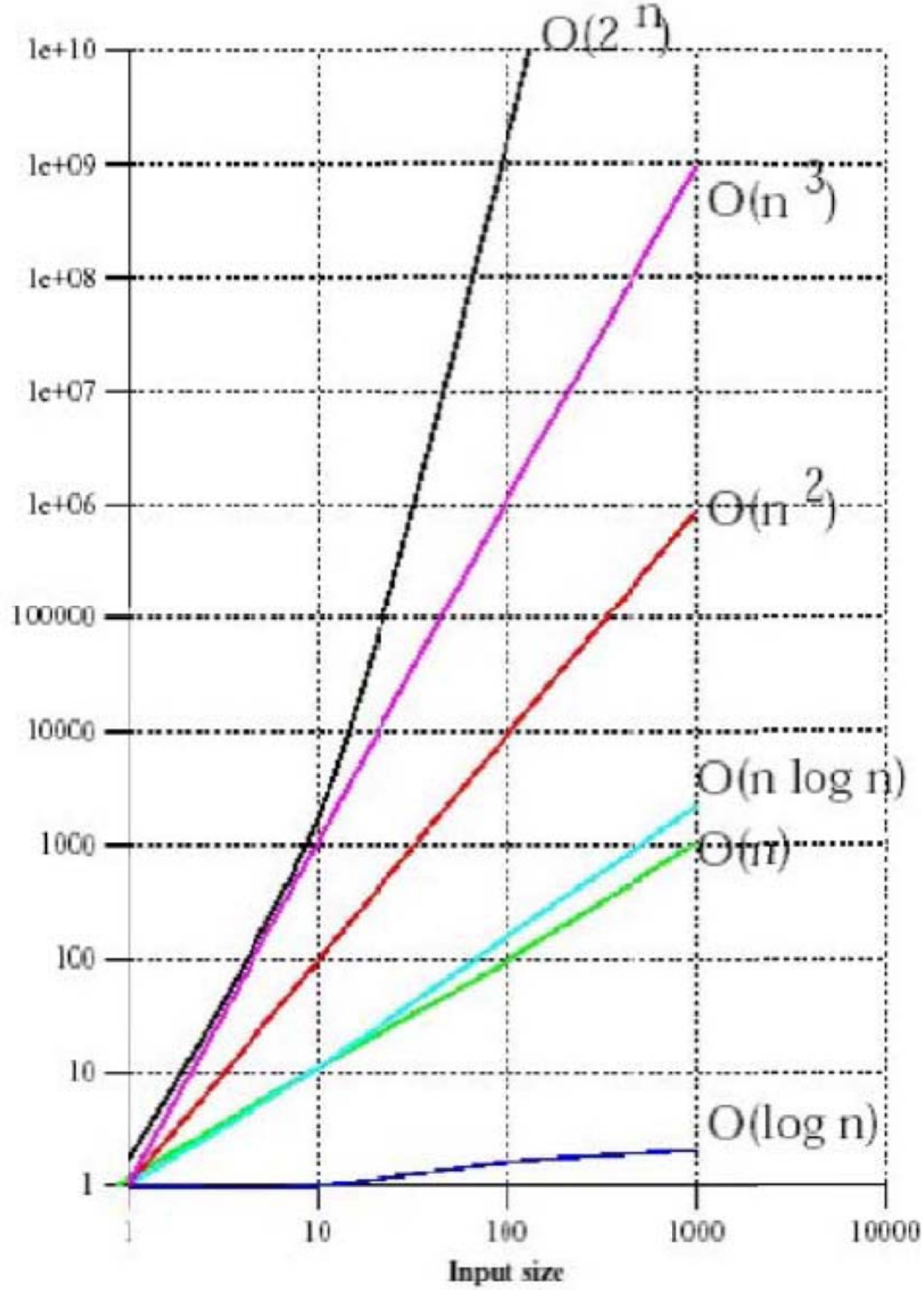
Formally: $f(n) = \Theta(g(n))$ if and only if:

$$f(n) = O(g(n))$$

and

$$f(n) = \Omega(g(n))$$

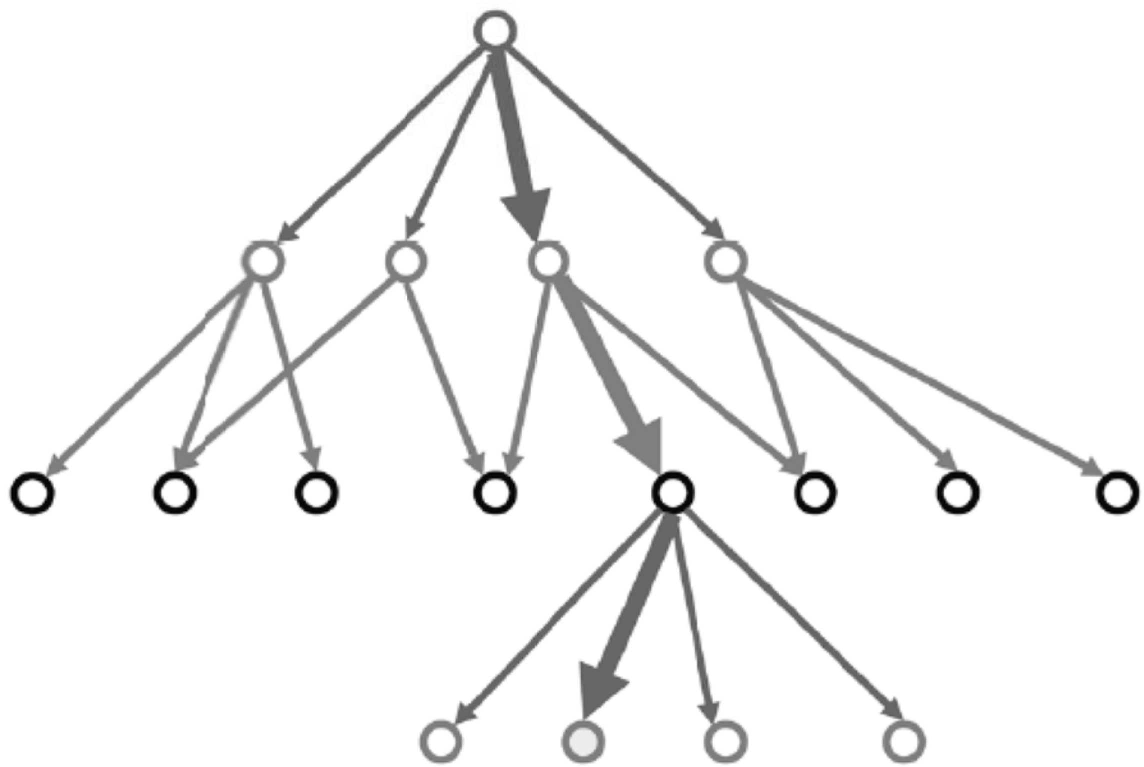
We then say `f is of order g’



O(2 n)
O(n 3)
O(n 2)
O(n log n)
O(n)
O(log n)
1e+10
1e+09
1e+08
1e+07
1e+06
100000
10000
1000
100
10
1
1
10
100
1000
10000
Input size

Recap: Game Verifiers

Given a path through a game, can you check if it is a valid winning path in polynomial time?



Oracles

Determinism vs Non-Determinism

Example Problem:

Is `x=96` part of the list `A=(9,4,55,67,...,96,8)` ?

Deterministic Algorithm

```
• for i=1 to n {
    if (A(i) = x) return true;
  }
  return false;
```

Non-deterministic Algorithm

```
• j ← oracle_choice (1:n)
  if {A(j) = x} return true
   else return false;
```

Deterministic Machines...

running on a particular input will always produce the same output ... and the underlying machine model (e.g. a Turing Machine) will always pass through the same pre-determined sequence of states based on the input data and program.

Non-deterministic Machines (with an oracle) ...

can `guess correctly’ at decision points and their output can be verified quickly by checking the produced certificate in polynomial time.

A decision problem is in P ...

if and only if it can be decided by a deterministic polynomial time Turing Machine.

A decision problem is in NP ...

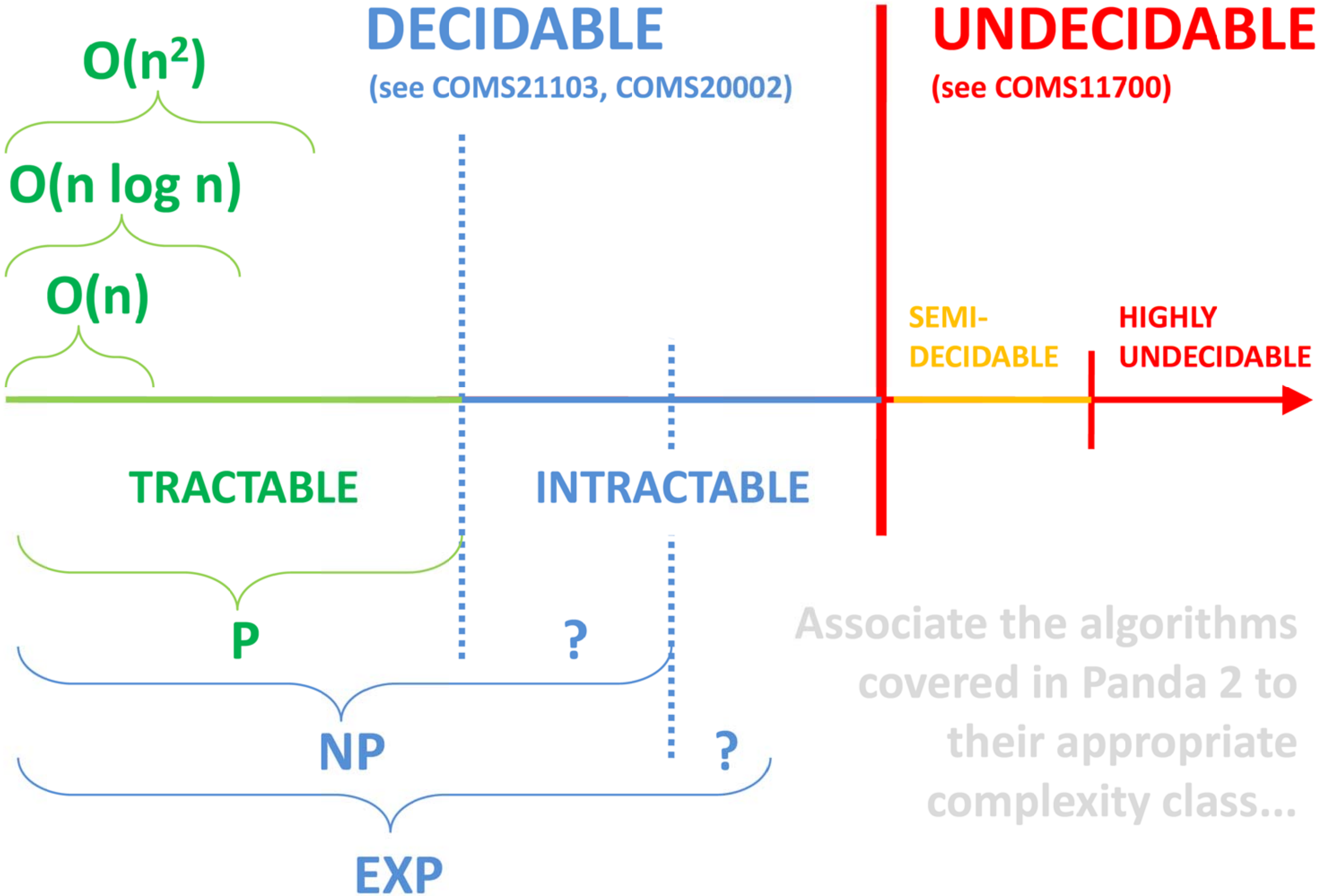
if and only if it can be decided by a non-deterministic polynomial time Turing Machine

A language is in NP ...

if and only if it has a polynomial time verifier. That is, there is a certificate which can check that a string is part of the language in polynomial time.

Problem Complexity Overview

University of BRISTOL
Department of Computer Science



Associate the algorithms covered in Panda 2 to their appropriate complexity class...