C53COM – Computability

Spring Semester 2007

Dr. Dario Landa-Silva

jds@cs.nott.ac.uk
www.cs.nott.ac.uk/~jds/teaching/g53com.html

LECTURE 1 – INTRODUCTION

Overview

- Aim of the Module
- Module Contents
- Format of Lectures
- Reading List
- Assessment
- Computability

Aim of the Module

The overall aim of the course is to provide an understanding of the concept of computability. In particular, course objectives are:

- To provide an appreciation of the many attempts that have been made to define the class of computable functions and also to indicate that all the sensible attempts to do this have been shown to be equivalent.

- To introduce the fact that there are problems that cannot possibly be solved on a computer.

- To show how, apparently different, computable problems are related in respect of their algorithmic complexity.

To achieve an understanding of the theory of computation, which is the basis for the design, analysis and efficient implementation of algorithms.

Module Contents

- Introduction
- Problems and Algorithms
- Mathematical Preliminaries Part 1
- Mathematical Preliminaries Part 2
- Turing Machines Part 1
- Turing Machines Part 2
- Special Turing Machines Part 1
- Special Turing Machines Part 2
- Solvability and Unsolvability
- More Undecidability
- Algorithm Analysis
- Complexity Classes
- Reductions and Completeness Part 1
- Reductions and Completeness Part 2
- The NP-Complete Class Part 1
- The NP-Complete Class Part 2
- Proving NP-Completeness Part 1
- Proving NP-Completeness Part 2
- Summary and Revision Part 1
- Summary and Revision Part 2

Format of Lectures

- Monday 12-1 pm in C1-JBCEXGE, Tuesday 10-11 am in A24-JBCSOUTH.
- The notes for the lectures will be available from the following url: www.cs.nott.ac.uk/~jds/teaching/g53com.html
- Students are also expected to make good notes in class.
- Exercises and extra notes will be available for some lectures.

Reading List

- Computational complexity. Christos H. Papadimitriou. Addison-Wesley, 1994.
- Computers and intractability: a guide to the theory of NP-completeness. Michael R. Garey, David S. Johnson. W. H. Freeman, 1979.
- For other suggested books, see the module reading list at: http://www.nottingham.ac.uk/is/gateway/readinglists/

Assessment

- Written Examination (100%), reasoning questions style. Previous years papers are available for reference at: http://www.nottingham.ac.uk/is/gateway/exampapers/

Computability

Computability focuses on answering questions such as:

- What class of problems can be solved using a computational algorithm?
- Is there an efficient computational algorithm to solve a given problem?

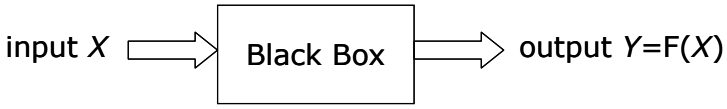
There have been many attempts by computer scientists to define precisely what is an effective procedure or computable function.

Intuitively, a computable function can be implemented on a computer, whether in a reasonable computational time or not.

An ideal formulation of an effective procedure or computable function involves:

1. A language in which a set of rules can be expressed.
2. A rigorous description of a single machine which can interpret the rules in the given language and carry out the steps of a specified process.

A very simplistic model of a computer is a black box that takes an input *X* and computes some output Y=F(X). The exact definition of function F(X) depends on the program.



Computability is concerned with formulating and understanding the modelling of the black box, its programs and the functions that can be effectively computable.

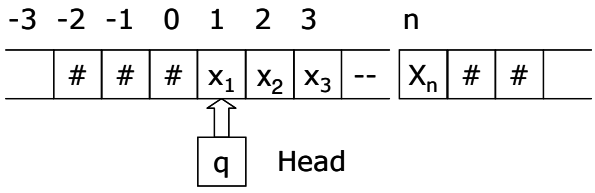
An effectively computable function is any function that can be computed in some machine with no limits on storage capacity.

Several models of the 'black box' computing machine have been proposed in the field of computer science:

- Turing machine (Turing, 1936)
- Post machine (Post, 1936)
- URIM – Unlimited register ideal machine (Sheperdson and Sturgis, 1963)

The Church's thesis (Church, 1936) expresses that 'every effectively computable function is Turing computable, i.e. is computable on a Turing machine'.

The Turing Machine (TM) is a computing model with no limits in storage capacity:



Components of the Turing machine are: locations in the tape, symbols of the alphabet, head, states, initial state, final states, and the program.

A step in a TM program is an action which is specified as a 5-tuple of the form:

(current state, current tape symbol, new state, new symbol, direction of move)

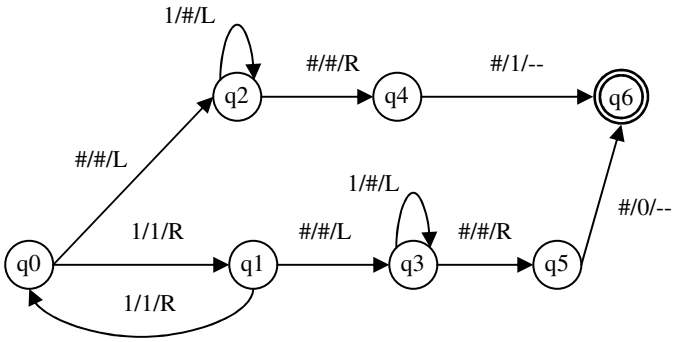
A Turing machine is deterministic if there is only one action or 5-tuple for any current (no final) state q_k and any current symbol that can occur when the machine is in state q_k.

Functioning of the Turing machine:

- Limitless storage modelled as a tape divided into locations
- A symbol from some prescribed alphabet can be written in each location
- Initially, each location in the tape is blanc (represented by the # symbol)
- The input $x=x_1x_2…x_n$ is written one symbol at a time in locations 1,2,…,n
- The head moves forward and backwards writing and deleting symbols
- The head can be in any one of a finite number of states.
- At the start of computation, the head will be in an initial state
- The machine only halts if the head reaches a final state
- During computation, the head will be in some state q_k and reading the current symbol in the tape
- If q_k is not a final state, the head can overwrite the current symbol in the tape, possibly move to the left or right position in the tape, and the state q_k may or may not change.
- When the machine halts, the output is taken from the location 1,2,…,m where m+1 is the first location greater than 0 which contains a blank symbol.

An example of a Turing Machine is given by the following program where q_0 is the initial state and q_6 is the final state.

$(q_0,1,q_1,1,R)$
$(q_1,1,q_0,1,R)$
$(q_0,\#,q_2,\#,L)$
$(q_1,\#,q_3,\#,L)$
$(q_2,1,q_2,\#,L)$
$(q_3,1,q_3,\#,L)$
$(q_2,\#,q_4,\#,R)$
$(q_3,\#,q_5,\#,R)$
$(q_4,\#,q_6,1,--)$
$(q_5,\#,q_6,0,--)$



The set of actions in a Turing machine program can also be represented as a two-dimensional array or as labelled directed graph.

What is the processing of inputs: x=1011, y=111, z=1111 by the Turing machine given above? What is the purpose of that Turing machine?

"Computational complexity is the area of computer science that contemplates the reasons why some problems are so hard to solve by computers". (Papadimitriou, 1994).

"I see complexity as the intricate and exquisite interplay between *computation* and (complexity classes) and *applications* (that is, problems)". (Papadimitriou, 1994).

Many problems are recognised to be inherently intractable which means that no algorithm can solve them quickly.

The theory of NP-completeness is concerned with intractable problems and provides techniques to prove that a given problem is intractable.

"In short, the primary application of the theory of NP-completeness is to assist algorithm designers in directing their problem=solving efforts towards these approaches that have the greatest likelihood of leading to useful algorithms". (Garey&Johnson, 1979)