

COMPUTABILITY AND INTRACTABILITY (2008-2009)

GUIDE TO REVISION

Generally speaking, exam questions will consist of a certain amount of book work together with some related rider(s); these might be interleaved to some extent. A student who has understood the material should be able to pass the exam without difficulty.

In your answers it is usually a good idea to give a brief indication of your overall plan before embarking on technical details (of course this does not apply to straightforward book-work). This way I can give you credit for relevant observations even if you do not get everything right. Of course no credit can be given for observations that are not relevant or just wrong; there is also no merit in sheer verbosity.

You might be asked to design a Turing machine to carry out a very simple task. In such a case give the relevant alphabets etc. and the instructions (as quintuples). The point of such a question is to give you the opportunity to show that you understand the basics by applying them to a simple task. In other situations you might need to describe a Turing machine that carries out a more complicated task (e.g., as part of a reduction). For such a situation focus on describing the machine at a fairly high level (essentially specifying its behaviour) taking care of any special cases. See the notes for examples. For these you do not need to go into low level detail.

Note that you will not be asked to invent complicated reductions; any that do occur will be fairly straightforward. One possibility is that a slightly longer reduction is split into two or more simple sub-parts, thus giving you both a structure for your answer and hints on what to do. It would however be reasonable to describe a more complicated reduction as part of a question and ask you to prove that it is correct. In such a case take time to understand what it does, e.g., if the reduction builds a graph for each given instance then draw some diagrams to see how it works. This will give you an idea of how to begin proving its correctness. Focus on clarity of expression rather than overly technical notation that you do not fully understand — of course if you do fully understand it and it is correct then that is fine! Past exam papers are a reasonable guide, but bear in mind that there have been some changes to the material.

Here is a brief guide to examinable topics.

Note 1: The important parts are Sections 1.2, 1.3 and 1.4. Section 1.5 is useful just as a clarification of the central notion of diagonalization insofar as it relates to ideas in computability. All other parts of this note are essentially general background and are not examinable. Understanding these sections means to be able to reason at this high level about the notion of computability; examples were covered in the lectures.

Note 2: This is just background and does not contain any directly examinable material; it is evidence in support of Turing's thesis (which you *must* know and be able to discuss if asked).

Notes 3-13: All the contents of these are examinable *except* for Note 5, *Random Access Machines* which can be omitted. Above all make sure you understand the key ideas (as described in lectures) and avoid the pitfalls, e.g., doing a reduction the wrong way round (all too common!) or 'proving' that something is unsolvable just by outlining one possible approach and observing that this fails! (Usually questions will be phrased in such a way as to prompt your memory.)

Note 14: This is not examinable.

Appendices A, B, C: These are not examinable, although it would be perfectly feasible to use simple ideas from from Appendices B and C in parts of questions.

Exercises: The methods and knowledge gained from the three exercise sheets are part of the course (but don't bother memorizing new definitions introduced there, if any). It would be quite reasonable for part of a question to be a variant of something seen in the exercises.