COM2003: Automata Logic and Computation (First Semester)

Guide to Examinable Material

Lecture	Concepts, Knowledge and Skills	Sipser
1	• Definition of DFAs • Definition of regular languages	31-43
2	• How to design simple DFAs using diagrams, and using transition functions • Definition of regular operations on regular languages • How to show that regular languages are closed under regular operations *union* and *complement*	44-47
3	• Definition of NFAs • How to design simple NFAs using diagrams, and using transition functions • How to show that regular languages are closed under the *concatenation* operation	47-53
4	• How to convert NFAs to DFAs • How to show that regular languages are closed under the *star* operation	54-62
5	• Definition of regular expressions • How to describe simple languages using simple regular expressions • How to construct NFAs and DFAs that recognise languages described by simple regular expressions	63-69
6	• Definition of GNFAs • How to convert a DFA to a GNFA and reduce to a single regular expression describing the language the machine recognises	69-76
7	• Definition of the pumping lemma for regular languages • How to apply the pumping lemma to show (non)regularity of a language	77-82
8	• Definition of CFGs • How to draw a parse tree for a string using a CFG	101-107
9	• Definition of ambiguous grammars • Definition of inherently ambiguous languages • Definition of Chomsky Normal Form • Definition of relationship between string length and parse-tree size for CFGs in Chomsky Normal Form	107-110
10	• Definition of PDAs • How to design simple PDAs using diagrams, and using transition functions	111-116
11	• How to design a PDA for a language, given a CFG for that language	117-125
12	• Definition of the pumping lemma for CFLs • How to apply the pumping lemma to show a language is (non)context-free	125-129
13	• Definition of a deterministic Turing Machine	139-

	• Definition of Turing-recognisable languages • Definition of Turing-decidable languages	147
14	• How to give a formal, implementation, or high-level description of a Turing Machine	150-156, 168-172
15	• How to show that the Halting Problem is undecidable • Definition of countable sets • Definition of uncountable sets • How to show uncountability by using diagonalisation • How to show that some problems are not Turing-recognisable • Definition of the language hierarchy from regular languages up to Turing-unrecognisable languages	167, 175-184
16	• Definitions of complexity notation • Definition of complexity class **EXPTIME** • Definition of complexity classes **P** and **NP** • Definition of example problems in **P** and **NP**	251-272
17	• Definition of known relationship between **P** and **NP**, and the two possibilities as to the actual relationship • Definition of polynomial time reducibility • Definition of *SAT* and *3SAT* • Definition of **NP-Complete** and example thereof	273-287, 290-294
18	• Definition of **NP-Hard** • Definition of Savitch's Theorem • Definition of **PSPACE** • Conjectured relationship between classes **P**, **NP**, **PSPACE** and **EXPTIME**	302, 307-313

Guidance:

- 'Definition' means be able to reproduce and apply the definition in a question.
- 'How to' means be able to apply a procedure in answering a question.
- Page references are to Sipser M., Introduction to the Theory of Computation (2^{nd} /international edition)

In addition, evidence of a general awareness of the context of computational models and results about them may be examined, for example with respect to applications of different models or languages in real-world computing.

Highest marks on questions will be reserved for *creative* applications of results and definitions covered during the course, to solve previously unseen problems.