

3515ICT: Theory of Computation

1 Summary

1.1 Background

- Sets, functions, strings, languages.
- Finite, countably infinite and uncountably infinite sets.
- Proofs that $\mathbf{N} \times \mathbf{N}$, \mathbf{Q} , Σ^* for Σ finite, ... are all countably infinite.
- Proofs by case analysis, construction, equivalence, contradiction, induction, diagonalisation.

1.2 Regular languages

- Reading and writing regular expressions, DFAs and NFAs.
- Language defined by a regular expression; language recognised by a DFA or NFA.
- Transformation of regular expressions into equivalent NFAs.
- Transformation of NFAs into equivalent DFAs, by subset construction algorithm.
- Transformation of DFAs into equivalent regular expressions, by state elimination algorithm, using GNFA's.
- Transformation of DFAs into equivalent minimal-state DFAs, by state equivalence algorithm.
- The regular languages are closed under concatenation, union, (Kleene) closure, intersection, complement, reversal, ...
- Decidable properties of regular languages: membership, emptiness, finiteness (and hence infiniteness), equality to Σ^* , equality, inclusion ($A \subseteq B$), ...
- Algorithms to demonstrate these properties are decidable.
- Proving languages are not regular, using the pumping lemma for regular languages.
- Examples of regular and nonregular languages.

1.3 Context-free languages

- Reading and writing context-free grammars and PDAs.
- Derivations, leftmost derivations, parse trees.
- Ambiguity of CFGs (and inherent ambiguity of CFLs). Transformation of ambiguous CFGs into equivalent unambiguous CFGs.

- Transformation of CFGs into Chomsky normal form.
- Transformation from CFGs into equivalent PDAs.
- Transformation from PDAs into equivalent CFGs.
- Deterministic PDAs. Every deterministic CFL (a CFL recognised by a deterministic PDA) has an unambiguous CFG.
- Proofs that every regular language is context-free.
- The CFLs are closed under concatenation, union, closure, reversal, intersection with a regular language, but not intersection and complement. Counterexamples.
- Decidable properties of CFLs: membership, emptiness, finiteness (and hence infiniteness), . . .
- Algorithms to demonstrate these properties are decidable, *e.g.*, derivation and dynamic programming algorithms for membership.
- Undecidable properties of CFLs: ambiguity, equality to Σ^* , equality, inclusion, empty intersection, . . .
- Proving languages are not context-free, using the pumping lemma for CFLs.
- Examples of context-free and non-context-free languages.

1.4 Computability

- Definition of deterministic and nondeterministic Turing machines.
- Reading and writing (implementation-level descriptions of) Turing machines.
- Variants of Turing machines: simulating doubly-infinite tape TMs with singly-infinite tape TMs, simulating multitape TMs with single-tape TMS, simulating NTMs with TMs.
- The Church-Turing thesis: every effectively computable function (resp., effectively recognisable language) can be computed (resp., recognised) by a Turing machine.
- Decidable and Turing-recognisable languages.
- Proof that every context-free language is decidable.
- The decidable languages are closed under concatenation, union, closure, intersection, complement, reversal, . . .
- The Turing-recognisable languages are closed under concatenation, union, closure, intersection, reversal, but not complement. Counterexample.
- Proof by diagonalisation that not all languages are Turing-recognisable.
- Proof by diagonalisation that the acceptance problem for TMs (A_{TM}) is not decidable (but is Turing-recognisable).

- Proof by diagonalisation that the halting problem for TMs ($HALT_{TM}$) is not decidable (but is Turing-recognisable).
- Reduction and its properties.
- Proof by reduction from A_{TM} that $HALT_{TM}$ is not decidable.
- Proof that the complement of A_{TM} , $\overline{A_{TM}}$, is not Turing-recognisable.
- Proof that $\overline{HALT_{TM}}$ is not Turing-recognisable.
- Proof that by reduction from A_{TM} that the emptiness problem for TMs (E_{TM}) is undecidable.
- Proof that by reduction from E_{TM} that the equality problem for TMs (EQ_{TM}) is undecidable.
- Proof that the nonemptiness problem for TMs (NE_{TM}) is Turing-recognisable, and hence E_{TM} is not Turing-recognisable.
- Rice's Theorem: Every nontrivial semantic property of TMs is undecidable.
- Applications of Rice's Theorem: Emptiness, finiteness, regularity, context-freeness, equality with Σ^* , equality, inclusion, closure under reversal, ... are all undecidable.
- The Post Correspondence Problem (PCP) is undecidable.
- Proof by reduction from PCP that CFG ambiguity is undecidable.
- Proof by reduction from PCP that CFG equality with Σ^* , equality, empty intersection, ... are all undecidable.
- Examples of decidable, undecidable and non-Turing-recognisable languages.

1.5 Complexity

- Definition of the class \mathcal{P} , and its properties. Examples of problems in \mathcal{P} .
- Definitions of the class \mathcal{NP} , and its properties. Examples of problems in \mathcal{NP} .
- Definition and properties of polynomial(-time) reducibility.
- Definitions of \mathcal{NP} -completeness, and its significance.
- How to prove a problem is \mathcal{NP} -complete?
- Meaning and significance of "satisfiability (SAT) is \mathcal{NP} -complete".
- Difference between SAT, CSAT and 3SAT.
- Proof that 3SAT is \mathcal{NP} -complete by polynomial reduction from CSAT.
- The Davis-Putnam algorithm for deciding satisfiability of CNF expressions.
- Facts: 2SAT and HORN SAT are in \mathcal{P} .

- Proofs by reduction from 3SAT that MAX2SAT and NAESAT are \mathcal{NP} -complete.
- Proofs by reduction from 3SAT that CLIQUE, Subgraph Isomorphism, Independent Set and Vertex Cover are \mathcal{NP} -complete.
- Proofs that Directed Hamiltonian Path, Undirected Hamiltonian Path and Travelling Salesman Problem are \mathcal{NP} -complete.
- Proof that Subset Sum is \mathcal{NP} -complete.
- Definition and properties of $\text{co}\mathcal{NP}$, PSPACE and EXP(TIME).
- Examples of $\text{co}\mathcal{NP}$ -complete problems.
- Examples of PSPACE-complete problems.
- Examples of EXP-complete problems.
- Managing \mathcal{NP} -complete problems.

References

- L. Fortnow, *Foundations of Complexity*. URL: <http://weblog.fortnow.com/> (See My Links → Foundations of Complexity)
- D. Harel with Y. Feldman, *Algorithmics: The Spirit of Computing, Third Edition*, Addison-Wesley, 2004.
- J.E. Hopcroft, R. Motwani and J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation, Second Edition*, Addison-Wesley, 2001.
- C.H. Papadimitriou, *Computational Complexity*, Addison-Wesley, 1994.
- M. Sipser, *Introduction to the Theory of Computation, Second Edition*, Course Technology, 2006.