Languages, Machines, and Classical Computation
Second Edition
Luis M. Augusto
## Contents

Preface to the 1st edition ........................................... xv
Preface to the 2nd edition ......................................... xix

1  Introduction  .................................................. 1
   1  Classical computation: Turing, von Neumann, and Chomsky  3
   1.1 Computers, information, and computations .................... 3
   1.2 Computational problems, algorithms, and decisions .......... 4
   1.3 The Turing-von Neumann paradigm ............................ 5
   1.4 Models of classical computation: Automata .................. 9
   1.5 The Chomsky hierarchy ....................................... 10

II  Preliminaries and notation ..................................... 15
   2  Mathematical notions ....................................... 17
      2.1 Basic notions .......................................... 17
      2.1.1 Sets, relations, functions, and operations ............ 18
      2.1.2 Binary relations and ordered sets .................... 26
      2.2 Discrete structures ..................................... 33
      2.2.1 Algebraic structures and operations ................... 34
      2.2.1.1 Algebras and morphisms .......................... 34
      2.2.1.2 Boolean algebras ................................ 35
      2.2.2 Graphs and trees ...................................... 40
      2.3 Proof techniques ........................................ 45
      2.3.1 Mathematical and structural induction ................. 46
      2.3.2 Proof by contradiction .............................. 47

III  Languages, machines, and classical computation ........ 51
   3  Formal grammars and languages .............................. 53
Contents

3.1 Basic notions ........................................................................... 54
  3.1.1 Strings and operations on strings .................................. 54
  3.1.2 Formal languages and operations thereon ...................... 56
  3.1.3 Formal grammars .......................................................... 59
    3.1.3.1 Central notions ....................................................... 59
    3.1.3.2 Rules, symbols, and grammar cleaning ................. 61
3.2 Regular languages .................................................................. 69
  3.2.1 Regular expressions ....................................................... 69
  3.2.2 Regular grammars ........................................................ 74
  3.2.3 Properties of regular languages ..................................... 80
    3.2.3.1 Pumping lemma for regular languages .................. 80
    3.2.3.2 Algebra and linear equations for regular languages .... 81
    3.2.3.3 Closure properties of the regular languages .......... 83
3.3 Context-free languages ......................................................... 88
  3.3.1 Context-free grammars ................................................... 88
    3.3.1.1 Context-free vs. context-sensitive grammars ........ 88
    3.3.1.2 Normal forms for CFGs I: Chomsky normal form .... 90
    3.3.1.3 Normal forms for CFGs II: Greibach normal form .... 94
  3.3.2 Properties of the context-free languages ......................... 105
    3.3.2.1 Pumping lemma for CFLs and Ogden’s lemma ......... 105
    3.3.2.2 Further properties of CFLs ................................. 108
3.4 Recursively enumerable languages ....................................... 116
3.5 The Chomsky hierarchy (I) ................................................... 121

4 Models of computation ......................................................... 125
  4.1 Finite-state machines ......................................................... 125
    4.1.1 Finite automata ....................................................... 126
      4.1.1.1 Basic aspects of finite automata ......................... 127
      4.1.1.2 Characteristic equations ................................. 133
      4.1.1.3 The pumping lemma for regular languages ......... 135
      4.1.1.4 The Myhill-Nerode theorem and FA minimization .... 136
      4.1.1.5 Deterministic and non-deterministic FAs ............. 142
      4.1.1.6 Kleene’s theorem and the properties of L ............ 149
Contents

4.1.2 Finite transducers ........................................ 153
4.1.2.1 Moore and Mealy machines .................. 153
4.1.2.2 Equivalence of finite transducers ............... 158
4.1.2.3 Minimizing finite transducers .................. 160
4.1.2.4 Conversion of transducers into acceptors ....... 165

4.2 Pushdown automata ........................................ 174
4.2.1 Basic aspects of PDAs .................................. 175
4.2.2 Two acceptance modes by PDAs: Final state and empty stack ........................................ 178
4.2.3 Equivalence between CFLs and PDAs .................. 180
4.2.4 CFLs accepted by deterministic PDAs ................. 187
4.2.4.1 Deterministic PDAs .................................. 187
4.2.4.2 LR(1) grammars .................................... 189

4.3 Turing machines ........................................... 204
4.3.1 Basic aspects of Turing machines .................... 204
4.3.2 Turing machines computing functions ................. 207
4.3.3 Turing machines accepting languages ................. 209
4.3.3.1 Turing machines and unrestricted grammars .................. 209
4.3.3.2 Linear-bounded automata: Special Turing machines for CSGs ................................ 213
4.3.4 The universal Turing machine ......................... 214

4.4 The Chomsky hierarchy (II) ................................. 220

5 Computability and complexity ................................. 225
5.1 The decision problem and Turing-decidability ........ 225
5.2 Undecidable problems and Turing-reducibility .......... 228
5.3 The Chomsky hierarchy (III) ............................... 234
5.4 Computational complexity .................................. 236
5.4.1 Computational problems ............................... 236
5.4.2 The Blum axioms and complexity measures .......... 237
5.4.3 Complexity classes ..................................... 241
5.4.4 The Cook-Levin theorem and polynomial-time reducibility ................................ 246
5.5 The Chomsky hierarchy (IV) ............................... 258

Bibliography ..................................................... 261

Index ............................................................. 267
# List of Figures

1.5.1 The basic postulate of the Chomsky hierarchy ........................................ 12
1.5.2 Two derivation trees ................................................................................. 14

2.1.1 A partially ordered set ............................................................................. 29
2.1.2 Hasse diagram of a poset .......................................................................... 33
2.2.1 A simple graph with five vertices and seven edges ................................. 42

3.2.1 A labeled digraph $\mathcal{G}(r)$ for a regular expression $r$ ...................... 75
3.2.2 A labeled digraph $\mathcal{G}(G)$ corresponding to a left-linear grammar $G$ .............................................................. 78
3.2.3 The digraph $\mathcal{G}'(G')$ obtained from $\mathcal{G}(G)$ ................................. 79
3.3.1 Derivation tree of the string $w = acbac \in L(G)$ with the corresponding partial derivation trees ..................................................... 101
3.3.2 Two leftmost derivations of the string $a^4 + a^3 a$ .................................. 102
3.3.3 Parse tree of an unambiguously derived string ....................................... 104
3.3.4 Parse trees for productions $(1) S \rightarrow a$ and $(2) S \rightarrow AB$ ........ 106
3.3.5 Parse tree for $z = uv^4w^2y$ .................................................................. 107
3.4.1 A derivation graph of the string $bab$ generated by a UG ...................... 119

4.1.1 Computer model of a FA ........................................................................... 127
4.1.2 State diagrams of FAs ............................................................................. 130
4.1.3 A FA with two accepting states and one rejecting state ......................... 130
4.1.4 A FA for the regular language $L = \{c, ba\}^* \{ac, aab\}$ ......................... 131
4.1.5 A finite automaton $M$ for the pumping lemma ..................................... 136
4.1.6 A FA (1) and its minimal equivalent FA (2) ....................................... 141
4.1.7 A N DFA for the language $L = \{001\}^* \{0, 010\}^*$ ............................ 143
4.1.8 Equivalent NDFAs with and without $\epsilon$-transitions ......................... 146
4.1.9 Equivalent N DFA (1) and FA (2) ........................................................... 150
4.1.10 Schematic diagrams for FAs accepting $(1) L_1 \cup L_2$, $(2) L_1 L_2$, and $(3) (L_1)^*$ ................................................................. 152
4.1.11 A FA accepting $L = L_1 \cup L_2$ .............................................................. 152
4.1.12 Moore (1) and Mealy (2) machines ....................................................... 155
4.1.13 A Mealy machine (1) and its equivalent Moore machine (2) ............ 161
4.1.14 A Mealy machine (1) and its minimal equivalent (2) ......................... 166
4.1.15 A Moore machine converted into a FA ................................................ 167
4.1.16 Deterministic finite automata ................................................................ 170
List of Figures

4.1.17 Mealy machines ........................................ 173
4.1.18 A barcode .................................................. 174
4.2.1 Computer model for a PDA ............................... 176
4.2.2 A PDA $M$ accepting the language $L(M) = \{a^m b^m | m \geq 0\}$ 178
4.2.3 Proving the equivalence of $L(M) = N(M)$ .................. 181
4.2.4 NDFA recognizing the viable prefixes for the CFG of Balanced Parentheses .............................. 194
4.2.5 Pushdown automata ........................................ 199
4.2.6 A PDA accepting $L(M) = \{u \in \Sigma^*|u = ww^R\}$ ........ 200
4.2.7 Top-down (1) and bottom-up (2) PDAs .................. 202
4.3.1 Computer model for a Turing machine .................. 205
4.3.2 A Turing machine that computes the function $f(m, n) = m + n$ for $m, n \in \mathbb{Z}^+$ .......................... 208
4.3.3 Turing machine $M_T$ that computes the function $f(m, n) = 2m + 3n$ for $m, n \in \mathbb{Z}^+$ .................. 210
4.3.4 Program for Turing machine $M_T$ that computes the function $f(m, n) = 2m + 3n$ for $m, n \in \mathbb{Z}^+$ .......... 211
4.3.5 The encodings $\langle M_T \rangle$ and $\langle M_T, z \rangle$ .......... 216
4.3.6 A combination of Turing machines ..................... 218
4.3.7 A Turing machine ........................................ 219
5.2.1 A combination of Turing machines ..................... 233
5.3.1 The Chomsky hierarchy and beyond: Decidable, Turing- recognizable, and not-Turing-recognizable languages .......... 235
5.4.1 The hierarchy of complexity classes with corresponding tractability status ........................................ 245
5.4.2 A tableau for the Turing machine $M$ ................... 251
5.4.3 Typical structure of $\text{NP}$-completeness proofs by polynomial-time reductions ................................. 254
## List of Tables

3.5.1 The Chomsky hierarchy. ........................................ 123

4.4.1 The extended Chomsky hierarchy: Grammars, languages, and associated computer models. ........................................ 222

5.3.1 Decidability (“Yes”) and undecidability (“No”) of some properties of interest for the Chomsky hierarchy. .................... 235

5.4.1 Rates of growth of some standard functions. .................. 242
List of Algorithms

3.1 Grammar cleaning .............................................. 63
3.2 Left-/Right-linear grammar to right-/left-linear grammar 79
3.3 Chomsky-normal-form transformation ........................ 92
3.4 Greibach-normal-form transformation ........................ 95
3.5 Language class by grammar type ............................ 124

4.1 Deterministic FA minimization ............................... 140
4.2 Subset-construction algorithm ............................... 148
4.3 Partition refinement for the states of a Mealy machine 163
4.4 Mealy machine minimization ................................. 164
4.5 Conversion of a CFG $G$ into a PDA $M$ ................. 184
Preface to the 1st edition

Teachers tend to be picky with the material they use in teaching contexts. This may be for personality reasons, but the variety of contexts and students also plays a role in this pickiness. Be it as it may, it often is the case that students end up with teaching material in many formats and from many different sources, creating often a lack of uniformity, both in notation and terminology. Because I am picky for all the reasons above, I typically feel that my teaching task is substantially facilitated and optimized when I have gone to the great lengths of putting all the material for a particular academic subject together in a single manual or textbook. This guarantees not only conceptional and notational uniformity, but also a selection of approaches that I feel work well, or better, for particular topics or problems.

This book is not about discovering the wheel; that is, possibly no novel contents are to be found in it. The objective when writing it was that of "putting together" a textbook on the classical theory of computing. If there is any novel aspect in this textbook, it may well be the fact that I insist on preceding the terms "(theory of) computation" and "(theory of) computing" with the adjective "classical" to collect under the same label the Chomsky hierarchy and the Turing-von Neumann paradigm of computing. The former comprises three closely associated central topics, to wit, formal grammars, formal languages, and models of computation (a.k.a. machines, or automata), and the latter gives to these, namely via the Turing machine, measures of the spatial and temporal costs of computation. I say that this collection constitutes (the) classical (theory of) computation, because many, often newer, other forms of computing have emerged or become (more) popular since the Turing "revolution," many of which today may be said to constitute the non-classical (theory of) computation. This is, for the initiated, more immediately the field of quantum computing, but other forms of computation such as artificial neural networks and evolutionary computing may be seen as also non-classical versions of computing.

It is arguably possible to produce a textbook on formal languages, grammars, and automata with no emphasis on computing, let alone with any specific computational concerns. One such approach might be with linguists in mind, though contemporary linguistics is not averse to
Preface to the 1st edition

...computation. On the extreme pole of this position, formal grammars, languages, and automata are often reduced to the theory of computation, namely as it serves the theoretical foundations of the digital computer. Without taking a reductive view, I discuss formal languages and grammars from the viewpoint of computation, and consider the associated automata as models thereof. This said, readers with other foci will find that the computational perspective taken here does not hinder—and may even facilitate—their particular interests and concerns.

The backbone of this book is undoubtedly the Chomsky hierarchy. Although much computing has run in the digital computer since N. Chomsky first conceived it, it still works well for combining the mostly linguistic approach with the computational one. In particular, it keeps reminding us that we are linguistic beings to the point that one of our most interesting creations—the digital computer—is language-based through and through, a feature well-patent in the famous Turing Test, a "test" conceived by the creator of the Turing machine to distinguish a human computer from a non-human one. Indeed, it seems to have been the rationale in Turing (1950) that language is sufficient to distinguish the human from the non-human computer or reasoner. More than anything, it might have been this insistence on the verbal behavior of computers that motivated the can-of-worms idea of AI (artificial intelligence) as ultimately aiming at human-like machines, at least from the viewpoint of intelligence, if not of emotion.

There is no way to go around this and it requires emphasis: (classical) computing is a mathematical subject. Although the presence of automata, of which the most famous is the Turing machine, lends it a flavor of engineering, these are not physical machines nor can they be; they are mathematical objects. To be sure, the digital computer is based on the Turing machine, but this has a feature—an infinite tape—that makes of the former a mere approximation of the latter. The mathematical nature of this subject accounts for the clearly mathematical approach in this book: I distinguish statements into definitions and propositions, and provide proofs (or sketches thereof) to further distinguished—if not distinct—statements, to wit, theorems and their companion lemmas and corollaries. The numbering of such statements finds its utility in internal referencing, if it gives a more high-brow quality to the main text. I reserve the status of theorems for statements of higher importance than propositions, but the reader is free to consider (most) propositions in this text as de-facto theorems; the fact that proofs are provided (or left as exercises) for propositions supports this view.

This mathematical nature of the subject also justifies the large selection of exercises here provided. Indeed, only few students are gifted
with mathematical skills that free them from the arduous and time-consuming practice of doing exercises. On the other hand, some may find this a pleasant activity. Between these fall most mortals, one should think. But the selection of exercises in this book was also guided by the belief that one should be confronted with novel material and problems, in order to develop research, as well as creative, skills.

Still with regard to the mathematical nature of this text, there are throughout it a few algorithms for the computation of specific functions (e.g., computing the Chomsky normal form of a given grammar). I chose not to stick to a single pseudo-code or to a single algorithm format in the belief that different algorithms can be better grasped in distinct ways. Yet another advantage of this might be the familiarity with diverse pseudo-codes and algorithm formats. Importantly, too, no programming language or software plays any role whatsoever in this book. This is so deliberately to keep the subject matter as general as possible, untied to specific implementations or applications.

As said above, the aim for this book is not (re)inventing the wheel. Although classical computing and its theory are in a current state of development, with many a problem as focus of research—notably so the \( P=\text{\#NP} \) problem—, the subject of the theory of classical computing has attained a certain fixed form that is historically justified. In the second half of last century, when this subject emerged, an abundance of textbooks and monographs were published, and a few of these established themselves as standard references in the field. As such, it is only natural that in pedagogical pursuits one should resort to them as sources. This I do with two such classics in particular, to wit, Davis & Weyuker (1983) and Hopcroft & Ullman (1979), the latter of which has evolved into the more undergraduate-friendly Hopcroft, Motwani, & Ullman (2013). A further source is Du & Ko (2001), a thoroughly mathematical approach. Readers can greatly benefit from a direct use of all these referenced works. Texts and manuals on this subject matter directed at undergraduate audiences abound, with many a good one to further assist readers in their academic pursuits. Referencing them all is of course impossible, but interested readers know where to find them. More specific, often more advanced, literature is cited throughout this text in the appropriate places; in particular, I cite the works in which important results (e.g., theorems) were first published.

Lastly, this textbook is a further elaboration on what was originally a chapter in a book of mine first published by College Publications, to wit, Augusto (2018). In this book, a chapter on the theory of computing appeared to be relevant, because issues such as Turing-completeness of logic programming and the complexity of the satisfiability problem
Preface to the 1st edition

(a.k.a SAT) required a minimal grasp of, among other topics, the Turing machine. Having resorted to this chapter to teach topics in automata, formal languages, and the classical theory of computation, and having obtained satisfactory results, I decided to expand it to what is now the present textbook. The main guideline for this expansion was the inclusion of topics that were left out in the mentioned chapter for spatial and temporal reasons, but which are essential for a fuller treatment of this subject. Some of these new topics—e.g., characteristic equations of finite automata, grammar cleaning algorithm—may appear quite inessential from an Anglo-Saxon perspective, but my individual work with Spanish students preparing themselves to take exams on the above-mentioned topics made me realize the need to be as encompassing and comprehensive as possible, namely with the large diversity of readers of this subject in mind.

I wish to thank Dov M. Gabbay, the scientific director of College Publications, and Ian Mackie, the editor for the Texts in Computing series, for publishing this book. My thanks go also to Jane Spurr, the managing director, for a smooth publication process.

Madrid, February 2019

Luis M. S. Augusto
Preface to the 2nd edition

The present edition corrects identified addenda and errata, and has both improved and new figures. It also includes the odd minor change in the main text of the first edition. The major changes are as follows:

- Figures 4.1.11 and 4.3.7 were replaced by more adequate ones: In the case of Figure 4.1.11, the union of languages was rather opaque, and the original finite automaton was replaced by a clearer illustration; as for Figure 4.3.7, the original Turing machine, which was too simple, was replaced by a more complex one.

- Exercise 4.1.14 has now five items, an increase aiming at providing more practice of a complex algorithm.

- An additional algorithm for the conversion of a context-free grammar into a pushdown automaton (Algorithm 4.5) is now included and, based on it, Example 4.2.3 was greatly revised and extended. This change entailed a new Figure (4.2.7), to be found in Exercise 4.2.8, which was completely redesigned.

These changes are now made available thanks to the readiness of College Publications to publish a second edition so shortly after the first. Renewed thanks are in order.

Madrid, May 2020

Luis M. S. Augusto


Index
Index

A
Adequateness, Structural, 104
Algorithm, CYK, 259
Alphabet, 54
Arden’s lemma, 82
Artificial neural network, 4
Automaton, Cellular, 4
Automaton, Finite (FA), 127
Automaton, Linear-bounded (LBA), 213
Automaton, Non-deterministic Finite (N DFA), 142
Automaton, Pushdown (PDA), 175
Automaton, Two-stack pushdown, 221
Automaton, Two-way finite (2FA), 174

Class, Language, 58
Classifier, 126
CNF (Conjunctive normal form), 38
Complement (of a language), 58
Complexity classes, 241
Complexity, Computational, 241
Complexity, Space, 238
Complexity, Time, 239
Computation, 3
Computation, Classical, 5
Computation, Evolutionary, 4
Computation, Fuzzy, 4
Computation, Non-classical, 5
Computation, Quantum, 4
Computational intelligence, 4
Computer, 3
Computer, Digital, 3
Computing, Hard, 4
Computing, Soft, 4
Concatenation, Language, 58
Concatenation, String, 55
Conjunctive normal form, Perfect, 38
Conversion: Mealy into Moore, 158
Conversion: Moore into Mealy, 158
Cook-Karp Thesis, 244
Cook-Levin Theorem, 249

De Morgan’s laws, 36
Index

Derivation, 59
Derivation graph, 117
Derivation, Direct, 59
Derivation, Leftmost, 59
Derivation, Rightmost, 59
Diagonalization method, 21
Digraph corresponding to a grammar, 77
Digraph corresponding to a regular expression, 72
Disjunctive normal form, Perfect, 38
DNF (Disjunctive normal form), 39
DPDA (Deterministic pushdown automaton), 187

E
Entscheidungsproblem, 6
Equivalence, Strong, 103
Equivalence, Structural, 103
Equivalence, Weak, 103

F
FA, Computation for a, 127
FA, Computer model for a, 127
FA, Configuration for a, 127
FA, Product, 172
FA, State diagram of a, 128
FA, Transition function of a, 127
FA, Transition table of a, 129
Finite-state machine, 125
Finite-state recognizer, 127
Finite-state transducer (FT'), 153
Frequency (of a letter), 54
FT (Finite-state transducer), 153
FT, Computation for a, 155
FT, Computer model for a, 154
FT, Configuration for a, 154

G
GNF (Greibach normal form), 94
Grammar equivalence, 60
Grammar, Ambiguous, 103
Grammar, Clean, 61
Grammar, Context-free (CFG);
  Type-2, 88
Grammar, Context-sensitive (CSG);
  Type-1, 90
Grammar, Formal, 59
Grammar, Left-linear, 74
Grammar, Linear, 74
Grammar, LL(k), 204
Grammar, LR(k), 189
Grammar, Phrase-structure, 121
Grammar, Regular; Type-3, 74
Grammar, Right-linear, 74
Grammar, S-restricted left-/right-
  linear, 77
Grammar, Transformational-generative, 11
Grammar, Unrestricted (UG); Type-
  0, 116

H
Hardware, 8
Hilbert's Tenth Problem, 234

I
Induction, Mathematical, 46
Induction, Structural, 46
Information, 3
Intersection, Language, 58

J
JFLAP [free software], 125

K
Kleene closure, 58
Kleene star, 58

L
Language, Context-free (CFL), 88
Language, Context-sensitive (CSL), 90
Index

Language, Decidable, 228
Language, Formal, 56
Language, Leftmost, 60
Language, Recursive, 220
Language, Recursively enumerable (REL), 116
Language, Regular, 70
Language, Rightmost, 60
Language, String, 115
Language, Tree, 115
LBA (Linear-bounded automaton), 213
LBA, Configuration of a, 213
Length (of a string), 54
Letter, 54

M
Mealy machine, 153
Mirror image (of a language), 57
Mirror image (of a string), 55
Monoid, Free, 57
Moore machine, 153
Myhill-Nerode Theorem, 138

N
Name of a rule, 117
N DFA (Non-deterministic finite automaton), 142
N DFA, Computer model of a, 143
Non-terminal (symbol), 59
Normal form, Chomsky (ChNF), 90
Normal form, Conjunctive (CNF), 38
Normal form, Disjunctive (DNF), 39
Normal form, Greibach (GNF), 94

O
Ogden’s Lemma, 108

P
P =? NP, 243
Palindrome, 55
Parser, LL(*), 204
PDA (Pushdown automaton), 175
PDA, Bottom-up, 200
PDA, Computation for a, 176
PDA, Computer model of a, 175
PDA, Configuration for a, 176
PDA, State diagram of a, 178
PDA, Top-down, 183
PDA, Transition table of a, 177
PDA, Two-way (2PDA), 203
Positive closure, 58
Post’s Correspondence Problem, 234
Power, i-th (of a language), 58
Power, i-th (of a string), 55
Precedence properties, 71
Prefix, 55
Problem for Horn formulas, The satisfiability (HORN-SAT), 258
Problem for quantified Boolean formulas, The satisfiability (QBF-SAT), 258
Problem, Computational, 236
Problem, Decision, 225
Problem, Function, 237
Problem, The 2-SAT, 246
Problem, The Acceptance (ACPT), 229
Problem, The Busy Beaver, 234
Problem, The Circuit Satisfiability (CIRCUIT-SAT), 247
Problem, The Clique (CLIQUE), 247
Problem, The Graph Colorability, 247
Problem, The Graph Isomorphism, 248
Problem, The Halting (HALT), 271
Index

229
Problem, The Hamiltonian Cycle (HAM-CYCLE), 247
Problem, The Hamiltonian Path (HAMPATH), 237
Problem, The k-SAT, 247
Problem, The Maximum Satisfiability (MAX-SAT), 258
Problem, The Relative Primes, 246
Problem, The Satisfiability (SAT), 248
Problem, The Shortest Path, 246
Problem, The State-Entry (STATE-ENTRY), 231
Problem, The Subgraph Isomorphism, 247
Problem, The Subset-Sum (SUBSET-SUM), 248
Problem, The Traveling Salesman (TSP), 248
Problem, The Vertex Cover (VERTEX-COVER), 247
Production rule, 59
Production, Copying, 62
Production, Empty, 62
Production, Recursive, 62
Production, Renaming, 62
Production, Right-recursive, 68
Production, Unit, 62
Proof by contradiction, 47
Proof, Constructive, 115
Pumping lemma for CFLs, 105
Pumping lemma for regular languages, 80, 135
Pushdown automaton (PDA), 175
Reductio ad absurdum, 47
Reduction, LR(k)-grammar, 190
Regular expression, 69
Reverse (of a language), 57
Reverse (of a string), 55
Rice's Theorem, 234
S
Savitch's Theorem, 243
Semi-decidability, 228
Semigroup, Free, 57
Sentential form, 59
Shuffle, Language, 58
Shuffle, String, 55
Software, 8
State, Trapping, 129
String, 54
String, Empty, 54
Substitution, 109
Substring, 55
Suffix, 55
Symbol, Accessible, 61
Symbol, Non-generating, 61
Symbol, Reachable, 61
Symbol, Well-defined, 61
Syntax, 56
T
Terminal (symbol), 59
Tractability, 244
Transducer, Finite (FT), 153
Transducer, Pushdown, 203
Transition function, Extended, 132
Transition relation, 142
Tree, Derivation, 99
Tree, Parse, 99
Truth-value assignment, 37
Turing machine, 204
Turing machine, Computation for a, 206
Turing machine, Computer model for a, 205
Index

Turing machine, Configuration of a, 205
Turing machine, Non-deterministic, 212
Turing machine, State diagram of a, 207
Turing machine, Total, 220
Turing machine, Transition table for a, 207
Turing machine, Universal, 214
Turing-decidability, 226
Turing-recognizability, 235
Turing-reducibility, 230
Turing’s Theorem, 7
Turing-von Neumann paradigm, 8

U
Union, Language, 58

V
Variable, 59
Variable, Start, 59
von Neumann architecture, 7

W
Word, 54

Y
Yield, 59