

# Contents

<b>Preface</b>	<b>ix</b>
<b>JFLAP Startup</b>	<b>xiv</b>
<b>1 Finite Automata</b>	<b>1</b>
1.1 A Simple Finite Automaton . . . . .	1
1.1.1 Create States . . . . .	2
1.1.2 Define the Initial State and Final State . . . . .	2
1.1.3 Creating Transitions . . . . .	2
1.1.4 Deleting States and Transitions . . . . .	3
1.1.5 Attribute Editor Tool . . . . .	3
1.2 Simulation of Input . . . . .	5
1.2.1 Stepping Simulation . . . . .	5
1.2.2 Fast Simulation . . . . .	7
1.2.3 Multiple Simulation . . . . .	8
1.3 Nondeterminism . . . . .	9
1.3.1 Creating Nondeterministic Finite Automata . . . . .	9
1.3.2 Simulation . . . . .	10
1.4 Simple Analysis Operators . . . . .	12
1.4.1 Compare Equivalence . . . . .	13
1.4.2 Highlight Nondeterminism . . . . .	13
1.4.3 Highlight $\lambda$ -Transitions . . . . .	13
1.5 Alternative Multiple Character Transitions* . . . . .	13
1.6 Definition of FA in JFLAP . . . . .	14
1.7 Summary . . . . .	15
1.8 Exercises . . . . .	15
<b>2 NFA to DFA to Minimal DFA</b>	<b>19</b>
2.1 NFA to DFA . . . . .	19

2.1.1	Idea for the Conversion . . . . .	19
2.1.2	Conversion Example . . . . .	20
2.1.3	Algorithm to Convert NFA $M$ to DFA $M'$ . . . . .	22
2.2	DFA to Minimal DFA . . . . .	23
2.2.1	Idea for the Conversion . . . . .	23
2.2.2	Conversion Example . . . . .	24
2.2.3	Algorithm . . . . .	28
2.3	Exercises . . . . .	29
<b>3</b>	<b>Regular Grammars</b>	<b>31</b>
3.1	Grammar Editing . . . . .	31
3.2	Brute Force Parser . . . . .	33
3.3	Convert a Right-Linear Grammar to an FA . . . . .	34
3.3.1	Algorithm to Convert Right-Linear Grammar to FA . . . . .	37
3.4	Convert an FA to a Right-Linear Grammar . . . . .	37
3.4.1	Conversion . . . . .	38
3.4.2	Algorithm to Convert FA to Right-Linear Grammar . . . . .	40
3.5	Definition of Regular Grammar in JFLAP . . . . .	40
3.6	Summary . . . . .	40
3.7	Exercises . . . . .	41
<b>4</b>	<b>Regular Expressions</b>	<b>45</b>
4.1	Regular Expression Editing . . . . .	45
4.2	Convert a Regular Expression to an NFA . . . . .	46
4.2.1	“De-oring” an Expression Transition . . . . .	47
4.2.2	“De-concatenating” an Expression Transition . . . . .	49
4.2.3	“De-staring” a Transition . . . . .	50
4.2.4	Surrounding Parentheses . . . . .	51
4.2.5	Automatic Conversion . . . . .	52
4.2.6	Algorithm to Convert RE to NFA . . . . .	52
4.3	Convert an FA to a Regular Expression . . . . .	53
4.3.1	Reforming the FA to a GTG . . . . .	53
4.3.2	Collapse Non-final, Non-initial States . . . . .	57
4.3.3	Regular Expression Formula . . . . .	59
4.3.4	Algorithm to Convert FA to RE . . . . .	59
4.4	Definition of RE in JFLAP . . . . .	60
4.5	Summary . . . . .	60

4.6	Exercises . . . . .	60
<b>5</b>	<b>Pushdown Automata</b>	<b>63</b>
5.1	A Simple Pushdown Automaton . . . . .	63
5.1.1	Building the NPDA . . . . .	64
5.1.2	Simulation of Input . . . . .	64
5.2	Nondeterministic PDA . . . . .	65
5.3	Definition of NPDA in JFLAP . . . . .	68
5.4	Summary . . . . .	69
5.5	Exercises . . . . .	70
<b>6</b>	<b>Context-Free Grammars</b>	<b>72</b>
6.1	Creating and Parsing Context-Free Grammars . . . . .	72
6.1.1	Entering a CFG in JFLAP . . . . .	72
6.1.2	Parsing with the Brute-Force Parser . . . . .	73
6.1.3	Parse Tree . . . . .	74
6.1.4	How Brute Force Parsing Works . . . . .	75
6.2	Converting CFG to NPDA . . . . .	77
6.2.1	Conversion . . . . .	77
6.2.2	Algorithm . . . . .	79
6.3	Converting NPDA to CFG . . . . .	79
6.3.1	Conversion . . . . .	80
6.3.2	Algorithm . . . . .	85
6.4	Summary . . . . .	85
6.5	Exercises . . . . .	86
<b>7</b>	<b>Transforming Grammars</b>	<b>88</b>
7.1	Removing $\lambda$ -Productions . . . . .	88
7.1.1	The Transformation . . . . .	89
7.1.2	Algorithm for Removing $\lambda$ -Productions. . . . .	91
7.2	Removing Unit-productions . . . . .	91
7.2.1	The Transformation . . . . .	91
7.2.2	Algorithm for Removing Unit-Productions. . . . .	94
7.3	Removing Useless Productions . . . . .	94
7.3.1	The Transformation . . . . .	94
7.3.2	Algorithm for Removing Useless Productions. . . . .	96
7.4	Converting CFG to CNF . . . . .	96
7.4.1	The Transformation . . . . .	97

7.4.2	Algorithm for Converting a CFG to CNF. . . . .	98
7.5	Summary . . . . .	99
7.6	Exercises . . . . .	99
<b>8</b>	<b>LL and SLR Parsing</b>	<b>101</b>
8.1	Getting Started on Parse Tables - FIRST and FOLLOW Sets . . . . .	101
8.1.1	FIRST sets . . . . .	101
8.1.2	Definition of FIRST . . . . .	104
8.1.3	FOLLOW sets . . . . .	104
8.1.4	Definition of FOLLOW . . . . .	106
8.2	LL(1) Parsing . . . . .	107
8.2.1	Parsing a string with the LL(1) parser. . . . .	107
8.2.2	Comparing with Corresponding NPDA . . . . .	109
8.2.3	Building the LL(1) Parse Table . . . . .	110
8.2.4	Algorithm for Building LL(1) Parse Table . . . . .	110
8.2.5	When a CFG is not LL(1) . . . . .	111
8.3	SLR(1) Parsing . . . . .	111
8.3.1	Parsing a string with the SLR(1) Parser . . . . .	111
8.3.2	Conversion of CFG to NPDA using SLR(1) Method . . . . .	112
8.3.3	Using the SLR(1) Parse Table . . . . .	114
8.3.4	Constructing a DFA that Models the Parsing Stack . . . . .	116
8.3.5	Building the SLR(1) Parse Table . . . . .	119
8.3.6	An example grammar with a $\lambda$ -production . . . . .	120
8.3.7	An example grammar that is not SLR(1) . . . . .	122
8.4	Summary . . . . .	123
8.5	Exercises . . . . .	124
<b>9</b>	<b>Turing Machines</b>	<b>127</b>
9.1	One-Tape Turing Machines . . . . .	127
9.1.1	Building a Turing Machine . . . . .	127
9.1.2	Simulation of Input on Incomplete Machine . . . . .	129
9.1.3	Completing the Turing Machine . . . . .	130
9.1.4	Simulation of Input on Complete Machine . . . . .	131
9.2	Turing Transducers and the Multiple Simulator . . . . .	133
9.3	Multi-tape Turing Machines . . . . .	135
9.3.1	Nondeterminism and the Substring . . . . .	135
9.3.2	Universal Turing Machine . . . . .	138

9.4	Definition of $n$ -tape TM in JFLAP . . . . .	142
9.5	Summary . . . . .	142
9.6	Exercises . . . . .	143
<b>10</b>	<b>L-systems</b>	<b>146</b>
10.1	L-system Creation and Display . . . . .	146
10.1.1	Turtle Symbols . . . . .	147
10.1.2	Rewriting Rules . . . . .	148
10.1.3	Pushing and Popping Turtles . . . . .	150
10.1.4	Polygon . . . . .	152
10.1.5	Parameters . . . . .	153
10.1.6	Increment/Decrement Parameter Commands . . . . .	154
10.1.7	Stochastic Rules . . . . .	156
10.1.8	Mathematical Expressions . . . . .	157
10.1.9	Turtle Commands with Arguments . . . . .	159
10.1.10	3D L-systems . . . . .	159
10.1.11	Contextual Rules . . . . .	162
10.2	Definition of L-system in JFLAP . . . . .	163
10.3	Summary . . . . .	163
10.4	Exercises . . . . .	164
<b>11</b>	<b>Other Grammars in the Hierarchy</b>	<b>169</b>
11.1	Unrestricted Grammars . . . . .	169
11.1.1	Introduction . . . . .	169
11.1.2	Simple Grammar . . . . .	170
11.1.3	Complex Grammar . . . . .	171
11.1.4	Slow Complex Grammar . . . . .	173
11.2	Context-sensitive Grammars . . . . .	175
11.2.1	Simple Grammar . . . . .	176
11.2.2	Complex Grammar . . . . .	177
11.3	Summary . . . . .	178
11.4	Exercises . . . . .	179
<b>A</b>	<b>L-system Quick Reference</b>	<b>180</b>
A.1	Turtle Commands . . . . .	181
A.2	Parameters . . . . .	182

<b>B JFLAP .jff File Format</b>	<b>183</b>
B.1 Brief Review of XML Terminology . . . . .	183
B.2 Elements Common to All Structures . . . . .	184
B.3 Automaton . . . . .	185
B.3.1 Defining States . . . . .	185
B.3.2 Defining Transitions . . . . .	185
B.3.3 Tapes for Turing Machines . . . . .	186
B.4 Grammar . . . . .	187
B.5 Regular Expression . . . . .	187
B.6 L-System . . . . .	187