

# Contents

<b>Introduction</b>	<b>1</b>
<b>1 Abstract programming languages</b>	<b>7</b>
1.1 Syntax	8
1.2 A general language	13
1.3 Interleaving operational semantics	22
1.4 Equivalence notions	26
1.5 Reachable subsystems and quotients	33
1.6 Correspondence with CCS and TCSP	37
<b>2 Connections with formal language theory</b>	<b>54</b>
2.1 Terminating traces	54
2.2 The Turing power	58
2.3 Counters	62
2.4 Decidability questions	68
<b>3 Representation by finite automata</b>	<b>73</b>
3.1 Extended transition systems	75
3.2 Syntax-driven construction	76
3.3 The extended transition system for a term	91
3.4 Properties of the extended transition system of a term	95
3.5 Consistency	106
3.6 Finitely representable subsets	113
<b>4 Representation by finite and safe Petri nets</b>	<b>117</b>
4.1 Definitions and terminology of net theory	119
4.2 Syntax-driven construction	125
4.3 Consistency, definedness, and finiteness	134
4.4 Further properties of the constructed nets	144
<b>5 A remark on the representation by finite Petri nets</b>	<b>149</b>

<b>6 Representation by finite and strict predicate/transition nets</b>	<b>155</b>
6.1 Predicate/transition nets . . . . .	159
6.2 Syntax-driven construction . . . . .	168
6.3 Distributed operational semantics . . . . .	177
6.4 Distributed consistency . . . . .	185
<b>Conclusion</b>	<b>199</b>
<b>Appendix: Some proofs of theorems from chapter 1</b>	<b>201</b>
<b>Bibliography</b>	<b>206</b>
<b>Mathematical notations, abbreviations</b>	<b>212</b>
<b>Index</b>	<b>213</b>