Contents

1. Introduction
   1.1. Software quality via formal methods and testing
      1.1.1. Correct implementation via testing
      1.1.2. Correct implementation via proofs
      1.1.2.2. Testing
      1.1.2.3. Formal methods and testing
   1.2. X-machines
      1.2.1. Why X-machines
      1.2.2. X-machines - a blend of finite state machines and data structures
      1.2.3. Stream X-machines
   1.3. Thesis summary

2. X-machines - a computational model framework
   2.1. Computational models survey
      2.1.1. Automata
      2.1.2. Pushdown automata
      2.1.3. Turing machines
      2.1.4. Computational models as specification tools
   2.2. X-machines
   2.3. Straight-move stream X-machines.
      2.3.1. SMS X-machines as a basis for a specification and testing method
   2.4. (Generalised) stream X-machines
      2.4.1. Regular stack stream X-machines
      2.4.2. RStack S X-machines and 1-stack SMS X-machines acceptors are equivalent
      2.4.3. Stack stream X-machines with markers
      2.4.4. RStack S X-machines and MStack S X-machines are equivalent
      2.4.5. Assessment of the stream X-machine model
   2.5. Conclusions and further work

3. Stream X-machines
   3.1. (Generalised) stream functions
   3.2. Stream X-machine specification of a correlator
      3.2.1. First model
      3.2.2 Second model
      3.2.3. Third model
   3.3. Parallel and sequential composition of stream X-machines
3.3.1. Parallel composition 84
3.3.2. Sequential composition 87
3.4. Minimal stream X-machines and minimal coverings of stream X-machines 89
3.4.1. Some finite state machine theory 90
3.4.1.1. Morphisms 91
3.4.1.2. State machine minimality 92
3.4.2. Stream X-machine minimality 94
3.4.3. Minimal coverings 101

4. Testing 110
4.1. Background 110
4.1.1 Program based testing 111
4.1.2. Functional testing 112
4.1.2.1. The category-partition method 112
4.1.3. Theoretical testing 114
4.1.4. Finite state machine testing 115
4.1.4.1. Preliminary concepts 115
4.1.4.2. The state machine testing method 117
4.1.4.3. Construction of the test set and complexity 117
4.1.4.3.1. Transition cover 117
4.1.4.3.2. Characterisation set 119
4.1.4.3.3. Complexity 120
4.1.4.4. Upper bounds for the test size 121
4.1.4.5. Improvement in the test size 121
4.1.4.6. Limitations of the finite state machine testing 122
4.2. Stream X-machine testing 122
4.2.1. Theoretical basis for stream X-machine testing 123
4.2.2 The stream X-machine testing method 128
4.2.3. Test set construction 130
4.2.4. Complexity 133
4.2.5. An improvement in the test set size 135
4.2.6. Expected outputs 135
4.2.7. Imposing ’design for testing’ properties 135
4.2.8. Case study 141
4.2.8.1. Stream X-machine specification 141
4.2.8.2. Imposing the ’design for test’ conditions 148
4.2.8.3. The test set 150
4.2.8.4. Discussion 156
4.2.9. Generalised stream X-machine testing 156
4.2.10. (Generalised) stream X-module testing 157
4.2.11. Discussion and conclusions 168

5. Refinement of stream X-machines 159
5.1. Refinement-definitions 159
5.2. Implementation 178
5.3. Testing 178
5.3.1. ’Design for testing’ conditions 180
5.3.1.1. Simple refinement 180
5.3.1.2. Complete refinement 180
5.3.1.2.1. Enforcing the completeness condition. 181
5.3.1.3. Output-distinguishable refinement set. 184
5.3.2. Fundamental test function and the refinement testing theorem. 185
5.3.3. The refinement testing method 190
5.4. Stream X-machine specification of a word processor 192
5.4.1. Stream X-machine specification 194
5.4.2. The control machine 195
5.4.3. Refinement 202
5.4.3.1. The module Mo 202
5.4.3.2. The transfer functions zo and yo 202
5.4.3.3. Preparatory definitions 203
5.4.3.4. The module M1. 205
5.4.3.5. The transfer functions z1 and y1. 206
5.4.3.6. The module M2. 206
5.4.3.7. The transfer functions z2 and y2. 210
5.4.4. Further refinements. 210
5.4.5. Using stream X-machines (X-modules) for specifying the processing functions 210
6. Conclusions 213
6.1. X-machines - a computational model framework 213
6.2. X-machines - a basis for a specification language. 213
6.3. Stream X-machines 214
6.4. Minimality and equivalence 214
6.5. Animation of executable (stream) X-machines 215
6.6. Testing and stream X-machines 216
6.7. Refinement 217
6.7.1. Refinement testing 218
6.8. Formal verification 218
Bibliography 220
Index of Notation and Concepts 224