

Java and the Java Virtual Machine

Robert Stärk, Joachim Schmid, Egon Börger

April 4, 2003

The contents of Jbook. This document is not for distribution. The Home-Page of Jbook is

<http://www.inf.ethz.ch/~jbook/>

where this document and more information about Jbook is available.

Contents

1. Introduction	1
1.1 The goals of the book	2
1.2 The contents of the book	3
1.3 Decomposing Java and the JVM	7
1.4 Sources and literature	11
2. Abstract State Machines	15
2.1 ASMs in a nutshell	15
2.2 Mathematical definition of ASMs	18
2.2.1 Abstract states	18
2.2.2 Transition rules and runs	22
2.2.3 Syntactic sugar	25
2.2.4 Exercises	26
2.3 Notational conventions	27
<hr/>	
Part I. Java	
<hr/>	
3. The imperative core $\text{Java}_{\mathcal{T}}$ of Java	33
3.1 Static semantics of $\text{Java}_{\mathcal{T}}$	33
3.1.1 Syntax of $\text{Java}_{\mathcal{T}}$	34
3.1.2 Type checking of $\text{Java}_{\mathcal{T}}$	37
3.1.3 Vocabulary of $\text{Java}_{\mathcal{T}}$	38
3.2 Transition rules for $\text{Java}_{\mathcal{T}}$	39
3.2.1 Expressions	40
3.2.2 Statements	41
3.2.3 Derived language constructs	44
3.2.4 Exercises	45
4. The procedural extension $\text{Java}_{\mathcal{C}}$ of $\text{Java}_{\mathcal{T}}$	47
4.1 Static semantics of $\text{Java}_{\mathcal{C}}$	47
4.1.1 Syntax of $\text{Java}_{\mathcal{C}}$	47
4.1.2 Class members	50
4.1.3 Interface members	52

4.1.4	Accessibility, visibility, hiding and overriding	53
4.1.5	Static type checking	58
4.1.6	Overloaded methods	58
4.1.7	Vocabulary of $\text{Java}_{\mathcal{C}}$	61
4.2	Transition rules for $\text{Java}_{\mathcal{C}}$	63
4.2.1	Exercises	66
5.	The object-oriented extension $\text{Java}_{\mathcal{O}}$ of $\text{Java}_{\mathcal{C}}$	71
5.1	Static semantics of $\text{Java}_{\mathcal{O}}$	71
5.1.1	Operators for reference types	72
5.1.2	Syntax of $\text{Java}_{\mathcal{O}}$	72
5.1.3	Constructor declarations	73
5.1.4	Field access expressions	74
5.1.5	Overloaded methods	75
5.1.6	Instance creation expressions	77
5.1.7	Type checking of $\text{Java}_{\mathcal{O}}$	78
5.1.8	Vocabulary of $\text{Java}_{\mathcal{O}}$	78
5.2	Transition rules for $\text{Java}_{\mathcal{O}}$	80
5.2.1	Exercises	83
6.	The exception-handling extension $\text{Java}_{\mathcal{E}}$ of $\text{Java}_{\mathcal{O}}$	87
6.1	Static semantics of $\text{Java}_{\mathcal{E}}$	87
6.1.1	Vocabulary of $\text{Java}_{\mathcal{E}}$	89
6.2	Transition rules for $\text{Java}_{\mathcal{E}}$	89
6.2.1	Exercises	93
7.	The concurrent extension $\text{Java}_{\mathcal{T}}$ of $\text{Java}_{\mathcal{E}}$	95
7.1	Static semantics of $\text{Java}_{\mathcal{T}}$	96
7.1.1	Vocabulary of $\text{Java}_{\mathcal{T}}$	96
7.2	Transition rules for $\text{Java}_{\mathcal{T}}$	98
7.2.1	Scheduling of multiple threads	101
7.2.2	Thread methods	103
7.2.3	Exercises	105
7.3	Thread invariants	106
8.	Java is type safe	111
8.1	Structural properties of Java runs	111
8.2	Unreachable statements	117
8.2.1	Exercises	119
8.3	Rules of definite assignment	121
8.3.1	Exercises	125
8.4	Java is type safe	126

Part II. Compilation of Java: The Trustful JVM

9. The JVM_{\mathcal{I}} submachine	139
9.1 Dynamic semantics of the JVM _{\mathcal{I}}	139
9.2 Compilation of Java _{\mathcal{I}}	142
10. The procedural extension JVM_{\mathcal{C}} of JVM_{\mathcal{I}}	147
10.1 Dynamic semantics of the JVM _{\mathcal{C}}	147
10.2 Compilation of Java _{\mathcal{C}}	153
11. The object-oriented extension JVM_{\mathcal{O}} of JVM_{\mathcal{C}}	155
11.1 Dynamic semantics of the JVM _{\mathcal{O}}	155
11.2 Compilation of Java _{\mathcal{O}}	157
12. The exception-handling extension JVM_{\mathcal{E}} of JVM_{\mathcal{O}}	159
12.1 Dynamic semantics of the JVM _{\mathcal{E}}	159
12.2 Compilation of Java _{\mathcal{E}}	163
13. Executing the JVM_{\mathcal{N}}	165
14. Correctness of the compiler	167
14.1 The correctness statement	167
14.1.1 The equivalence of <i>pos</i> and <i>pc</i>	169
14.1.2 The equivalence of <i>restbody</i> and <i>opd</i>	169
14.1.3 The equivalence of <i>locals</i> and <i>reg</i>	171
14.1.4 The equivalence of <i>frames</i> and <i>stack</i>	171
14.1.5 Equivalence of states	174
14.2 The correctness proof	178

Part III. Bytecode Verification: The Secure JVM

15. The defensive virtual machine	209
15.1 Construction of the defensive JVM	210
15.2 Checking JVM _{\mathcal{I}}	210
15.3 Checking JVM _{\mathcal{C}}	213
15.4 Checking JVM _{\mathcal{O}}	214
15.5 Checking JVM _{\mathcal{E}}	219
15.6 Checking JVM _{\mathcal{N}}	221
15.7 Checks are monotonic	222
16. Bytecode type assignments	223
16.1 Problems of bytecode verification	224
16.1.1 Why are subroutines problematic?	224
16.1.2 Why sets of reference types?	229
16.2 Successors of bytecode instructions	231
16.2.1 Successors for JVM _{\mathcal{I}} instructions	232
16.2.2 Successors for JVM _{\mathcal{C}} instructions	232

- 16.2.3 Successors for JVM_O instructions 233
- 16.2.4 Successors for JVM_E instructions 234
- 16.3 Type assignments without subroutine call stacks 236
- 16.4 Soundness of bytecode type assignments 242
- 16.5 Certifying compilation 252
- 17. The diligent virtual machine** 273
 - 17.1 Principal bytecode type assignments 273
 - 17.2 Verifying JVM_T 275
 - 17.3 Verifying JVM_C 279
 - 17.4 Verifying JVM_O 283
 - 17.5 Verifying JVM_E 283
 - 17.6 Verifying JVM_N 286
- 18. The dynamic virtual machine** 289
 - 18.1 Initiating and defining loaders 289
 - 18.2 Loading classes 290
 - 18.3 Dynamic semantics of the JVM_D 291
 - 18.3.1 Refinement of the run-time state 291
 - 18.3.2 Loading references 294
 - 18.3.3 The trustful JVM_D 298
 - 18.3.4 The defensive JVM_D 302
 - 18.3.5 The diligent JVM_D 303
 - 18.3.6 Exercises 303

Appendix

- A. Executable Models** 305
 - A.1 Overview 305
 - A.2 Java 306
 - A.2.1 Completing the Java ASM 307
 - A.2.2 Starting the machine 308
 - A.2.3 Introduction to the GUI 308
 - A.2.4 Running an example 310
 - A.2.5 Breakpoints 311
 - A.2.6 Examples on the CD 312
 - A.3 Compiler 312
 - A.3.1 Completing the Compiler ASM 312
 - A.3.2 Starting the machine 312
 - A.3.3 Introduction to the GUI 313
 - A.3.4 Examples on the CD 314
 - A.4 Java Virtual Machine 314
 - A.4.1 Completing the JVM ASM 315
 - A.4.2 Starting the machines 316

A.4.3	Introduction to the GUI.....	316
A.4.4	Examples on the CD.....	319
A.4.5	Dynamic Loading – Saraswat’s Example.....	319
B.	Java	323
B.1	Rules	323
B.2	Arrays	331
C.	JVM	335
C.1	Trustful execution	335
C.2	Defensive execution	343
C.3	Diligent execution	344
C.4	Check functions	347
C.5	Successor functions	348
C.6	Constraints	349
C.7	Arrays	351
C.8	Abstract versus real instructions.....	355
D.	Compiler	361
D.1	Compilation functions.....	361
D.2	maxOpd	363
D.3	Arrays	364
	References	365
	List of Figures	369
	List of Tables	373
	Index	375