



Universiteit
Leiden
The Netherlands

Abstract Behavioral Specification: unifying modeling and programming

Bezirgiannis, N.

Citation

Bezirgiannis, N. (2018, April 17). *Abstract Behavioral Specification: unifying modeling and programming*. IPA Dissertation Series. Retrieved from <https://hdl.handle.net/1887/61629>

Version: Not Applicable (or Unknown)

License: [Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden](#)

Downloaded from: <https://hdl.handle.net/1887/61629>

Note: To cite this publication please use the final published version (if applicable).

Cover Page



Universiteit Leiden



The following handle holds various files of this Leiden University dissertation:
<http://hdl.handle.net/1887/61629>

Author: Bezirgiannis, N.

Title: Abstract Behavioral Specification: unifying modeling and programming

Issue Date: 2018-04-17

Abstract Behavioral Specification: unifying modeling and programming

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Leiden
op gezag van de Rector Magnificus prof. mr. C. J. J. M. Stolker,
volgens besluit van het College voor Promoties
te verdedigen op dinsdag 17 april 2018
klokke 15.00 uur

door

Nikolaos Bezirgiannis
geboren te Thessaloniki, Griekenland,
in 1987

PhD committee

Promotor: Prof. dr. F.S. de Boer

Co-promotor: Dr. C. P. T. de Gouw Open Universiteit

Other members:

Prof. dr. A. Plaat

Prof. dr. F. Arbab

Prof. dr. E. B. Johnsen University of Oslo

Prof. dr. T. van der Storm Rijksuniversiteit Groningen



The work reported in this thesis has been carried out at the Center for Mathematics and Computer Science (CWI) in Amsterdam and Leiden Institute of Advanced Computer Science at Leiden University, under the auspices of the research school IPA (Institute for Programming research and Algorithmics). This research was supported by the European FP7-610582 project ENVISAGE on Engineering Virtualized Resources.

Contents

1	Introduction	1
1.1	Why ABS	4
1.2	Targetting Haskell	6
1.3	Validation	8
1.4	Outline	8
2	Background: the ABS Language	13
2.1	Data structures	14
2.2	Functional code	17
2.3	Side-effectful and OO code	19
2.4	Type system	21
2.4.1	Parametric Polymorphism	22
2.4.2	Subtype polymorphism	22
2.4.3	Variance	24
2.4.4	Type Synonyms	25
2.5	Module system	26
2.6	Metaprogramming with Deltas	26
2.7	Concurrency model	27
2.8	History of ABS	33
2.9	Comparison to other concurrent, modeling languages	33
3	HABS: A Variant of the ABS Language	35
3.1	Differences with Standard ABS	35
3.2	Language extensions to Standard ABS	38
3.2.1	Exceptions	38
3.2.2	Parametric type synonyms	41
3.2.3	Type Inference	42
3.2.4	Foreign Language Interface	42

3.2.5	Language extension for HTTP communication	44
3.3	Compiling ABS to Haskell	45
3.3.1	Compiler infrastructure	46
3.3.2	Functional code	47
3.3.3	Stateful code	48
3.3.4	Object encoding	51
3.3.5	Interfaces, Classes and Methods	52
3.4	Typing ABS	54
3.4.1	Subtyping	55
3.5	Runtime execution	59
3.6	Comparison to other ABS Backends	63
3.6.1	Comparing language support and features	63
3.6.2	Comparing runtime implementations	65
3.6.3	Benchmarking the ABS backends	66
3.7	Formal verification of HABS	68
3.7.1	Restricting to a subset of ABS	72
3.7.2	Operational Semantics	74
3.7.3	Target Language	77
3.7.4	Correctness	81
3.7.5	Resource Preservation	84
3.7.6	Experimental Evaluation	85
3.7.7	Proofs and auxiliary results	87
3.8	Case Study on Preferential Attachment	104
3.8.1	Results	105
3.9	Related Work	106
4	Resource-aware Modeling in HABS	111
4.1	Modeling time	111
4.2	Modeling virtualized hardware resources	113
4.3	Modeling systems	113
4.4	A real-time implementation	115
4.4.1	Comparison with symbolic-time execution	116
4.5	Case study: DevOps-in-the-Loop	118
4.5.1	The tool	122
4.5.2	Benchmark	124
4.6	Related Work	129

5	A Distributed Implementation of HABS	131
5.1	Implementation	133
5.1.1	Connection to Cloud infrastructure	134
5.1.2	Serialization	137
5.1.3	Garbage Collection	137
5.1.4	Failures in the Cloud	138
5.2	Extension: Service Discovery	139
5.3	Experiments and Results	141
5.4	Case Study: Distributed Preferential Attachment	143
5.5	Related Work	144
5.5.1	Distributed programming languages	145
5.5.2	Cloud middleware and management	145
6	Conclusion and Future Work	149
6.1	Future Work	150
	Summary	154
	Samenvatting	156
	Bibliography	159

