

Asynchronous Programming in the Abstract Behavioural Specification Language

Azadbakht, K.

Citation

Azadbakht, K. (2019, December 11). Asynchronous Programming in the Abstract Behavioural Specification Language. Retrieved from https://hdl.handle.net/1887/81818

Version:	Publisher's Version
License:	<u>Licence agreement concerning inclusion of doctoral thesis in the</u> <u>Institutional Repository of the University of Leiden</u>
Downloaded from:	https://hdl.handle.net/1887/81818

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

Cover Page



Universiteit Leiden



The handle <u>http://hdl.handle.net/1887/81818</u> holds various files of this Leiden University dissertation.

Author: Azadbakht, K. Title: Asynchronous Programming in the Abstract Behavioural Specification Language Issue Date: 2019-12-11

Propositions belonging to the PhD dissertation Asynchronous Programming in the Abstract Behavioural Specification Language

By Keyvan Azadbakht

- 1. Current programming means of parallelism and communication are insufficient for optimally exploiting the computational power of multicore architectures (Chapter 1).
- 2. It is generally recognized that asynchronous communication is well suited for distributed applications and therefore the actor model is a natural fit for such systems (Chapter 5).
- 3. The resulting integration of Actors with object-orientation allows for new object-oriented models of concurrency that are better suited for the analysis and construction of distributed systems than the standard model of *multi-threading* (Chapter 1, 4 and 5).
- 4. ABS can bridge modelling and programming for the purpose of formal reasoning and verification (this thesis).
- 5. Data streaming is gaining ever-growing applications in different domains. Integration of data streams with ABS enables formal reasoning and verification of data streaming in a concurrent object system (Chapter 4).
- 6. Absence of particular deadlocks caused by coroutines in an actor-based model of concurrency that features Cooperative Scheduling is decidable, even in the context of unbounded method invocations (Chapter 6).
- 7. The current challenge in language design is to efficiently use the underlying multicore resources while maintaining a high level of abstraction for the programmer.
- 8. Only the most capable programmers can explicitly control concurrency, and efficiently make use of the relatively small number of cores readily available today.