



Universiteit  
Leiden  
The Netherlands

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

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 <http://hdl.handle.net/1887/81818> 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

# Part III

## Enhancing Parallelism

This part consists of the following chapters:

**Chapter 4** Asynchronous Actor-based software programming has gained increasing attention as a model of concurrency and distribution. Many modern distributed software applications require a form of continuous interaction between their components which consists of streaming data from a server to its clients. In this chapter, we extend the basic model of asynchronous method invocation and return in order to support the streaming of data [13]. We introduce the notion of “future-based data streams” by augmenting the syntax, type system, and operational semantics of ABS. The application involving future-based data streams is illustrated by a case study on social network simulation.

**Chapter 5** In this chapter we introduce a new programming model of multi-threaded actors which feature the parallel processing of their messages [15]. In this model an actor consists of a group of active objects which share a message queue. We provide a formal operational semantics, and a description of a Java-based implementation for the basic programming abstractions describing multi-threaded actors. Finally, we evaluate our proposal by means of an example application.