SCIENCES & BIOENGINEERING SCIENCES

The Research Group

## Software Languages Lab (SOFT)

has the honor to invite you to the public defense of the PhD thesis of

## Quentin STIÉVENART

to obtain the degree of Doctor of Sciences

Title of the PhD thesis:

# Scalable Designs for Abstract Interpretation of Concurrent Programs: Application to Actors and Shared-Memory Multi-Threading

**Promotors:**
Prof. dr. Coen De Roover
Prof. dr. Wolfgang De Meuter

The defence will take place on

**Friday May 25 2018 at 16:00**

in Auditorium D.0.07 at the Campus Humanities, Sciences and Engineering of the Vrije Universiteit Brussel, Pleinlaan 2 - 1050 Elsene, and will be followed by a reception.

**Members of the jury:**

Prof. dr. Viviane Jonckers (chairman)
Prof. dr. Simon Keizer (secretary)
Prof. dr. Ann Dooms
Prof. dr. em. Theo D'hondt
Prof. dr. David Van Horn
(University of Maryland, USA)
Prof. dr. Philipp Haller
(KTH Royal Institute of Technology, Sweden)

## Curriculum vitae

Quentin Stiévenart obtained a "*Master en ingénieur civil en informatique*" degree at the ULB in 2014. He then joined the Software Languages Lab at the VUB to continue the research initiated in his Master's thesis. In the course of his PhD research, he co-authored 2 journal articles, 4 full conference papers and 2 short conference papers. He also presented his results at several international conferences and workshops. His research on static program analyses for concurrent programs is characterized by formalisations of the theory accompanied by practical implementations.

## Abstract of the PhD research


Concurrent programs are difficult for developers to reason about. They consist of concurrent processes, of which the execution can be interleaved in an exponential number of ways. Contemporary concurrent programs moreover feature dynamic process creation and termination, exacerbating the need for tool support.

Static program analysis can be a powerful enabler of such tool support, but current static analysis designs for concurrent programs are limited with respect to one or more of the following desirable properties: automation, soundness, scalability, precision and support for dynamic process creation. Moreover, existing analyses are designed with a single concurrency model in mind, and uniform design methods applicable to multiple concurrency models are lacking.

We study the applicability of a recent design method for static program analyses—abstracting abstract machines (AAM)—to concurrent programming languages. Applying this method results in analyses featuring all desirable properties except scalability. We present MacroConc and ModConc, two AAM-inspired uniform design methods that, when applied to the operational semantics of a concurrent programming language, result in static analyses featuring all the desired properties.

The first design method, MacroConc, introduces Agha's 1997 notion of macro-stepping into AAM analyses for concurrent programs. Refining the default all-interleavings semantics for concurrent processes with macro-stepping reduces the number of interleavings the analysis has to explore. The resulting analyses remain exponential in their worst-case complexity, but mitigate the scalability issues of existing analyses without compromising their precision. The second design method, ModConc, introduces Cousot and Cousot's 2002 notion of modularity into AAM analyses for concurrent programs. Analyses resulting from this design method consider each process of a concurrent program in isolation to infer potential interferences with other and newly created processes, which will have to be reconsidered until a fixed point is reached. Process interleavings are not explicitly modeled by the analysis but still accounted for. This analysis design trades off precision to yield process-modular analyses that scale linearly with the number of processes created in the program under analysis.

To demonstrate generality, we apply each design method to two prominent concurrent programming models: concurrent actors and shared-memory multi-threading. We prove the soundness and termination properties for each of the resulting analyses formally, and evaluate their running times, scalability and precision empirically on a set of 56 concurrent benchmark programs. Analyses resulting from the application of MacroConc achieve high precision, yet exhibit a reduction in running time of up to four orders of magnitude, compared to analyses resulting from a naive application of AAM. Analyses resulting from the application of ModConc exhibit a more consistent reduction, compared to both analyses resulting from the application of AAM and of MacroConc, but this at the cost of lower precision.

The complementary design methods presented in this dissertation enable one to select an analysis design fitting their needs in terms of scalability and precision, enabling future tool support for contemporary concurrent programs.