



Universiteit
Leiden
The Netherlands

Automata-theoretic protocol programming

Jongmans, Sung-Shik Theodorus Quirinus

Citation

Jongmans, S. -S. T. Q. (2016, March 3). *Automata-theoretic protocol programming*. *IPA Dissertation Series*. Retrieved from <https://hdl.handle.net/1887/38223>

Version: Corrected 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/1887/38223>

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

Stellingen

behorende bij het proefschrift
“Automata-Theoretic Protocol Programming”
van Sung-Shik T.Q. Jongmans

1. Today’s abstractions for protocol programming complicate writing correct protocol implementations, writing efficient/scalable protocol implementations, and writing modular protocol implementations. [Chapter 1]
2. Constraint automata constitute a suitable formalism for modeling protocols at a high level of abstraction and, as such, an appropriate semantics for a DSL for protocol programming; Reo and FOCAML constitute appropriate, complementary syntaxes for such a DSL. [Chapters 2, 3]
3. Constraint automata admit automatic compilation into increasingly efficient executables as more optimizations, modeled and proven correct at the high level of automata, are applied. [Chapters 4–8]
4. With sufficiently many optimizations in place, a DSL for protocol programming based on constraint automata can yield performance comparable to hand-written lower-level code, as shown in experiments with the NPB benchmarks. [Chapter 9]
5. Compositional priority can be expressed in a coloring model with a finite number of colors.
6. The relation between multiparty session types and exogenous coordination is insufficiently well-understood and should be studied more carefully to advance both subfields.
7. As much as possible, “boring” proofs should be formalized using proof assistant tools, or at least be amenable to automated proof checking.
8. For programmers to write high-performance code, a programming language can be simultaneously both inadequately low-level (for programmers to carefully control compute resources) and inadequately high-level (for a compiler to reconstruct programmers’ intentions necessary to automatically apply optimizations).
9. When you have an office with a sensor, instead of a switch, to control the lighting, be sure that the sensor reaches your desk.

10. Programming should *not* be a compulsory subject in primary education.
[<http://tweakers.net/nieuws/99114>]
11. Computer scientists and software engineers have a responsibility to make our profession be taken more seriously by society.
[<http://www.jasonbock.net/jb/News/Item/7c334037d1a9437d9fa6506e2f35eaac>]