PhD Thesis

Foundations Of Pure Functional Agent-Based Simulation

Jonathan Thaler (4276122)
jonathan.thaler@nottingham.ac.uk

supervised by
Dr. Peer-Olaf Siebers
Dr. Thorsten Altenkirch

September 21, 2018

Abstract

TODO

Contents

Chapter 1

Introduction

The traditional approach to Agent-Based Simulation (ABS) has so far always been object-oriented techniques, due to the influence of the seminal work of Epstein et al [1] in which the authors claim "[..] object-oriented programming to be a particularly natural development environment for Sugarscape specifically and artificial societies generally [..]" (p. 179). This work established the metaphor in the ABS community, that *agents map naturally to objects* [5] which still holds up today.

This thesis challenges that metaphor and explores ways of approaching ABS with the functional programming paradigm using the language Haskell. It is the first one to do so on a *systematical* level and develops a foundation by presenting fundamental concepts and advanced features to show how to leverage the benefits of it [3, 2] to become available when implementing ABS functionally. By doing this, the thesis both shows *how* to implement ABS purely functional and *why* it is of benefit of doing so, what the drawbacks are and also when a pure functional approach should *not* be used.

This thesis claims that the agent-based simulation community needs functional programming in ABS because of its *scientific computing* nature where results need to be reproducible and correct while simulations should be able to massively scale-up as well. The established object-oriented approaches need considerably high effort and might even fail to deliver these objectives due to its conceptually different approach to computing. In contrast, this thesis will show that by using functional programming for implementing ABS it is easy to add parallelism and concurrency, the resulting simulations are easy to test and verify, guaranteed to be reproducible already at compile-time, have few potential sources of bugs and are ultimately very likely to be correct.

Argument & Story of my Thesis with the components (publications) i have so far and which i still plan to do

1.1 Projectplan

2018 oct: begin publishing of STM paper oct & nov: writing of property paper december: begin publishing of property paper oct - dec: define structure, argument & story of thesis 2019 jan - sept: writing thesis, conducting additional research, finishing publication of STM and Testing papers. apr - july: writing and publishing "towards paper"

1.2 Publications

Throughout the course of the Ph.D. four (4) papers were published:

1. art of iterating (TODO cite): This paper derives the 4 different update-strategies and their properties possible in time-driven ABS and discusses them from a programming-paradigm agnostic point of view. It is the first paper which makes the very basics of update-semantics clear on a conceptual level and is necessary to understand the options one has when implementing ABS purely functional.

2. pure functional epidemics (TODO cite): establishing *how* to implement ABS in Haskell using non-monadic (Yampa) and monadic FRP

3. a tale of lock-free agents (TODO cite):

4. hands off my property (TODO cite):

5. towards pure functional agent-based simulation (TODO cite):

6. the equilibrium - totality correspondence (TODO cite): extended abstract

1.3 Contributions

1. This thesis is the first to *systematically* investigate the use of the functional programming paradigm, as in Haskell, to Agent-Based Simulation, laying out in-depth technical foundations and identifying its benefits and drawbacks. Due to the increased interested in functional concepts which were added to object-oriented languages in recent years because of its established benefits in concurrent programming, testing and software-development in general, presenting such foundational research gives this thesis significant impact.

2. This thesis is the first to show the use of STM in ABS and its potential benefit over lock-based approaches. STM is particularly strong in pure FP because of retry-semantics can be guaranteed to exclude non-repeatable persistent side-effects already at compile time. By showing how to employ STM it is possible to implement a simulation which allows

massively large-scale ABS but without the low level difficulties of concurrent programming, making it easier and quicker to develop working and correct concurrent ABS models. Due to the increasing need for massively large-scale ABS in recent years [4], making this possible within a purely functional approach as well, gives this thesis substantial impact.

3. This thesis is the first to present the use of property-based testing in ABS which allows a declarative specification- and hypothesis testing of the implemented ABS directly in code with *automated* test-case generation. This is an addition to the established Test Driven Development process and a complementary approach to unit-testing, ultimately giving the developers an additional, powerful tool to test the implementation on a more conceptual level. This should lead to simulation software which is more likely to be correct, thus making this a significant contribution with valuable impact.

4. This thesis is the first to outline the potential use of *dependent types* to Agent-Based Simulation on a *conceptual level* to investigate its usefulness for increasing the correctness of a simulation. Dependent types can help to narrow the gap between the model specification and its implementation, reducing the potential for conceptual errors in model-to-code translation. This immediately leads to fewer number of tests required due to guarantees being expressed already at compile time. Ultimately dependent types lead to higher confidence in correctness due to formal guarantees in code, making this a unique contribution with high impact.

1.4 Thesis structure

TODO: focus on strong narrative thus all chapters are interconnected and tell the story

1. time-driven vs. event-driven ABS 2. time-driven: start with update-strategies 3. present pure functional approach 4. how can we implement the 4 update-strategies in our approach 5. additional research: interaction between agents 6. additional research: event-driven ABS

leave dependent types for further research but write an additional short chapter on it which uses 2nd year report

Chapter 2

Conclusions

References

[1] Epstein, J. M., and Axtell, R. *Growing Artificial Societies: Social Science from the Bottom Up.* The Brookings Institution, Washington, DC, USA, 1996.

[2] Hudak, P., Hughes, J., Peyton Jones, S., and Wadler, P. A History of Haskell: Being Lazy with Class. In *Proceedings of the Third ACM SIGPLAN Conference on History of Programming Languages* (New York, NY, USA, 2007), HOPL III, ACM, pp. 12–1–12–55.

[3] Hudak, P., and Jones, M. Haskell vs. Ada vs. C++ vs. Awk vs. ... An Experiment in Software Prototyping Productivity. Research Report YALEU/DCS/RR-1049, Department of Computer Science, Yale University, New Haven, CT, Oct. 1994.

[4] Lysenko, M., and D'Souza, R. M. A Framework for Megascale Agent Based Model Simulations on Graphics Processing Units. *Journal of Artificial Societies and Social Simulation 11*, 4 (2008), 10.

[5] North, M. J., and Macal, C. M. *Managing Business Complexity: Discovering Strategic Solutions with Agent-Based Modeling and Simulation.* Oxford University Press, USA, Mar. 2007. Google-Books-ID: gRATDAAAQBAJ.