THE AGENTS NEW CLOTHS
TOWARDS PURE FUNCTIONAL PROGRAMMING IN ABS

Jonathan Thaler
Peer Olaf Siebers

School Of Computer Science
University of Nottingham
7301 Wollaton Rd
Nottingham, United Kingdom
{jonathan.thaler,peer-olaf.siebers}@nottingham.ac.uk

ABSTRACT

TODO

Keywords: Agent-Based Simulation, Functional Programming, Haskell, Concurrency, Parallelism, Property-Based Testing, Validation & Verification.

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 (Epstein and Axtell 1996) 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* (North and Macal 2007) which still holds up today.

In this paper we challenge this metaphor and explore ways of approaching ABS using the functional programming paradigm with the language Haskell. We present fundamental concepts and advanced features of functional programming and we show how to leverage the benefits of it (Hudak and Jones 1994, **?**) to become available when implementing ABS functionally.

We claim that the 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, we claim 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.

To substantiate our claims we implemented the *full* SugarScape (Epstein and Axtell 1996) model in Haskell as a case-study and discuss the implications, drawbacks and benefits of a pure functional implementation of it.

The aim and contribution of this paper is to introduce the functional programming paradigm using Haskell to ABS on a *conceptual* level, identifying benefits, difficulties and drawbacks. To the best of our knowledge, it is the first one to do so.

2 CASE-STUDY: PURE FUNCTIONAL SUGARSCAPE

TODO

why sugarscape - original sugarscape sparked ABS and use of OOP, therefore - quite complex model, will challenge implementation techniques

[1]

(**?**)

ACKNOWLEDGMENTS

The authors would like to thank

REFERENCES

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

Hudak, P., and M. Jones. 1994, October. "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.

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

Weaver, I. "Replicating Sugarscape in NetLogo". Technical report.

[1]The code is freely accessible from https://github.com/thalerjonathan/phd/tree/master/public/towards/code