Towards pure functional ABS Part I

The suitability of Haskell in ABS

Jonathan Thaler
School of Computer Science
University of Nottingham
jonathan.thaler@nottingham.ac.uk

Peer-Olaf Siebers
School of Computer Science
University of Nottingham
peer-olaf.siebers@nottingham.ac.uk

***Abstract*—So far, the pure functional paradigm hasn't got much attention in Agent-Based Simulation (ABS) where the dominant programming paradigm is object-orientation, with Java being its most prominent representative. In this paper we examine the suitability of the pure functional programming language Haskell for implementing ABS. We approach the problem from a general direction and look into the features and power of Haskell, compare it with existing technologies in ABS and see what potential benefits and drawbacks using Haskell in ABS are. The findings are directly applied in the second part, published as a separate paper, where we introduce a general-purpose library implemented in Haskell for implementing all kinds of ABS.**

***Index Terms*—Agent-Based Simulation, Haskell, Functional Programming**

I. Introduction

A. The power of a language

[ ] more expressive: we can express complex problems more directly and with less overhead. note that this is domain-specifix: the mechanisms of a language allow to create abstractions which solve the domain-specific problem. the better these mechanisms support one in this task, the more powerful the language is in the given domain. now we end up by defining what "better" support means [ ] one could in pronciple do system programming in haskell by provoding bindings to code written in c and / or assembly but when the program is dominated by calls to these bindings then one could as well work directly in these lower languages and saves one from the overhead of the bindings [ ] but very often a domain consists of multiple subdomains. [ ] my hypothesis is that haskell is not well suited for domains which are dominated by managing and manipulating a global mutable state through side-effects / effectful computations. examples are gui-programming and computer games (state spread accross GPU and cpu, user input,...). this does not mean that it is not possible to implememt these things in haskell (it has been done with some sucess) but that the solution becomes too complex at some point. [ ] conciesness [ ] low ceremony [ ] susceptibility to bugs [ ] verbosity [ ] reasoning about performance [ ] reasoning about space requirements

References