0.0.1 Functional Reactive Agent-Based Simulation

Report by:	Jonathan Thaler
Status:	Experimental, active development

Implementations of Agent-Based Simulation (ABS) have so far been reduced to the context of Object-Orientation (OO). We investigate how ABS can be implemented in a pure functional language like Haskell. The fundamental problem is that unlike in OO there are no objects and no implicit aliases through which to access and change data: method calls are not available in FP. We solve the problem of how to represent an agent and how agents can interact with each other. We build on the concept of Functional Reactive Programming for which we use the library Yampa. This allows us to represent agents as signal-functions with different types as input and output. In each time-step an agent gets fed in an input and creates an output which is the input for the next time-step, creating a feedback. The input- and output-types contain incoming and outgoing messages and various events like start, termination, kill. Also we build on the facilities Yampa provides for time-flow of a system be it continuous or discrete. For interactions between Agents we implemented messages, which are interactions over simulated time and a novel concept which we termed 'conversations', which are restricted interactions during which the simulation-time is halted. It is of most importance to us to keep our code pure - except from the reactive Yampa-loop all our code is pure and does not make use of the IO-Monad. Currently we are implementing the full *SugarScape* model of J.M.Epstein and R.Axtell as described in their book, which serves as the initial use-case to drive the implementation of our library. In the next step we also want to implement *Agent_Zero*, the most recent Agent-Model developed by J.M.Epstein. Further research will go into validation & verification of ABS in our library using QuickCheck and algebraic reasoning. The code is freely available but not stable as it currently serves for prototyping for gaining insights into the problems faced when implementing ABS in Haskell. The plan is to release the final implementation at the end of the PhD as a stable and full-featured library on Hackage. If you are interested in the on-going research please contact Jonathan Thaler (jonathan.thaler@nottingham.ac.uk).

Further reading

Repository: https://github.com/thalerjonathan/phd/tree/master/coding/libraries/frABS