# PhD Project-Plan

Jonathan Thaler

January 31, 2017

This document gives detailed information about the structuring of the research undertaken in this PhD.

## 1 Years

The whole PhD lasts for 3 years, 36 Months, from October 2016 to September 2019 and thus I will structure it according to 3 years where each year will be a major milestone - which is also intended by the Computer School.

### 1.1 1st Year: Groundwork

In this year I will learn basics and develop and research the methodology I will use for the main work in the 2nd year. Also I want to write a paper about this work and try to publish it on a Conference or Journal. The time-frame will be set by the date of the 1st year annual oral report which will happen beginning of July thus there are about 6 Months time (counting including January '17 and leaving out July '17). These are the things I want / need to achieve this year:

- Prototyping in Haskell, Scala and Java
- Study Actor-Model theory: Hewitt, Greif, Clinger, Agha
- Get into reasoning about programs
- Basics of Economics [2], [3]
- Basics of ACE: Tesfatsion
- Implement PureAgents Library
- Write & publish papers
- Write 1st year report
- Prototype ACE using Akka and Haskell

Important and mile-stones:

| | |
|---|---|
| End of March | Finished and submit Paper |
| June (Mid) | Finished writing 1st year report |
| July | Oral annual report |

#### 1.1.1 January

**Research** Finish implementation work in Java and Haskell

**Paper** First draft

**Reading**

- Book "Gleichzeitige Ungleichzeitigkeiten"
- Complex Systems Stuff
- Find semantics of simulation papers

#### 1.1.2 February

**Research**

- Theoretical: Specification language for ABS in Haskell.
- Programming: ABS and Yampa.

**Paper** Refine paper: read by supervisors and colleagues

**Reading**

- Re-Read existing papers on functional programming and simulations
- Read Semantics of Actors (Greif & Clinger)
- ACE papers

#### 1.1.3 March

**Research**

- Theoretical: Specification language for ABS in Haskell.
- Programming: ABS and Yampa.

**Reading**

- Read Semantics of Actors (Greif & Clinger)
- ACE papers

**Paper** Finalize and submit

#### 1.1.4 April

**Research** ABS and Yampa

**Paper** Write potential conference papers on "specification language" and "reactive agents"

**Reading** ACE papers

#### 1.1.5 May

**Paper** Continue writing potential conference papers on "specification language" and "reactive agents"

**Reading** Read papers not read so far but include in 1st year report

**Writing** Start writing 1st year report

#### 1.1.6 June

**Research** Do 1st Year oral exam

**Reading** Read papers not read so far but include in 1st year report

**Writing** Finalize 1st year report

#### 1.1.7 July to September

**Research** Prototyping ACE using my Haskell-Library and Akka: bilateral trading, EDSL for ACE and qualitative modelling, reasoning in ACE

**Reading**

- ACE basics papers (Tesfatsion)
- ACE models for trading/bartering (e.g. Gintis)

#### 1.1.8 October

2nd year starts

## 1.2 2nd Year: Main Work

Applying 1st year results, methods and experiences to develop and write main paper to be published in a journal in 3rd year thus in 2nd year the main work and implementation will be done. The idea is to start from Ionescus Framework [**?**] and build on his paper.

- Implement Ionescous framework using the methodology developed in 1st year
- Generalize implementation to market models
- Learn Agda and dependent types
- Dig deeper into equilibrium theory
- Get into market-microstructure: [**?**], [1]
- Dig into *emergent properties* of systems. Can they be formalized?
- Get into basics of Category-Theory [**?**] [4]
- Get into basics of Type-Theory (found good lectures on Youtube)

## 1.3 3rd Year: Finalizing, Publishing & Writing

I plan to be finished - or nearly finished - at the end of the 3rd year. In this year I will finalize the work of the 2nd year, publish the my main journal paper (and optional fun-papers if possible) and will write down the thesis.
To have a bit of distraction and to prevent myself to become too locked in in writing on the thesis I will also work on my optional fun-papers (see below) and hope to at least finish them and maybe publish them - at least I want to present them to 2-3 audiences (e.g. FP Lunch) to test the reaction (especially the Genesis-Paper).

- Finalize research of 2nd year
- Publish journal paper
- Write thesis
- Work on fun-papers

# 2 Papers

This is the list of papers I have in my mind and includes both work mandatory for the PhD and optional ones. The latter ones are just fun/philosophical-papers, not directly related to my PhD but somehow tangential with the very basic direction - they are intended to be worked on in my free-time and to free my head when wrestling too hard with my PhDs main work.

## 2.1 The Art of Iterating: Update-Strategies in Agent-Based Simulations

**Type:** Groundwork
**Target:** Conference/Journal
**Requirement:** Mandatory

When developing a model for an Agent-Based Simulation (ABS) it is of very importance to select the right update-strategy for the agents to produce the desired results. In this paper we develop a systematic treatment of all general properties, derive the possible update-strategies in ABS and discuss their interpretation and semantics something which is still lacking in the literature on ABS. Further we investigate the suitability of the three very different programming languages Java, Haskell and Scala with Actors to implement each of the update-strategies. Thus this papers contribution is the development of a new, general terminology of update-strategies and their implementation comparison in various kinds of programming languages.

## 2.2 Specification equals Code: An EDSL for pure functional Agent-Based Modelling & Simulation

**Type:** Groundwork
**Target:** Conference/Journal
**Requirement:** Mandatory

Building upon our previous work on update-strategies in Agent-Based Modelling & Simulation (ABM/S) where we showed that Haskell is a very attractive alternative to existing object-oriented approaches we ask in this paper if the declarative power of the pure functional language can be utilized to write specifications for simple ABM/S models which can be directly translated to our Haskell implementation.

## 2.3 Reactive Agents: Functional Reactive Programming and ABM/S

**Type:** Groundwork
**Target:** Conference/Journal

**Requirement:** Mandatory

In our previous work on update-strategies in Agent-Based Modelling & Simulation (ABM/S) we showed that Haskell is a very attractive alternative to existing object-oriented approaches but our presented approach was too limited and we hypothesized that embedding it within a functional reactive framework like Yampa would leverage it to be able to build much more complex models. In this paper we investigate whether this hypothesis is true by testing if our approach can easily be transferred to Yampa, what we really gain from it and if more complex models can become reality. As a proof-of-concept we build a large, complex model from Agent-Based Computational Economics (ACE) which simulates a whole economy.

### 2.4 Pure Functional ACE (Catchy title yet to be defined)

**Type:** Main PhD Work
**Target:** Journal
**Requirement:** Mandatory

Is the main work of the PhD and targeted at publication in a Journal. The exact topic and content will be clarified at the beginning of the 2nd year. Mainly it will describe how to implement Ionescus Framework of Gintis trading model and extend it to a more general Market-Model. It will also give an outlook on implementing it using dependent types.

### 2.5 Pure by Nature: A Library for pure Agent-Based Simulation & Modelling in Haskell

**Type:** Extension
**Target:** Conference
**Requirement:** Optional

This paper describes the ideas and theory behind the implementation of my ABM/S library "PureAgents" in Haskell.

### 2.6 Time in Games: a Tron Light-Cycle Game in Dunai

**Type:** Fun
**Target:** Conference
**Requirement:** Optional

This paper describes the 2D light-cycle game inspired by the movie Tron implemented in Dunai. It allows to turn back time.

## 2.7 Pure Functional Islamic Design

**Type:** Fun
**Target:** Conference
**Requirement:** Optional

Inspired by the paper "Functional Geometry" by Peter Henderson I had the idea to come up with a EDSL for declaratively describing pictures of islamic design which are then rendered using the gloss-library. From its focus totally unrelated to the PhD topic but still a great opportunity to learn Haskell, to learn to think functional, to learn to design my own EDSL - thus it may be a great paper to pursue even if I won't finish or produce something publishable.

## 2.8 The Genesis According to Computer-Science: Reality as Simulation of Free Will

**Type:** Philosophy
**Target:** ?
**Requirement:** Optional

I've always been interested in a deeper meaning behind things so I want to look into the philosophy and future of simulation: why do we simulate, what can we derive from simulations, what does it say that we humans simulate, what will the future of simulation be?
I claim that our ability to "simulate" in our mind separates our intelligence from those of the animals and that this is a unique property of humans. Also i think the future of simulation will be that humankind will do its own creation/live (artifical life, conciousness) which allows to accurately simulate a given setting - this of course could have ethical implications.

# 3 TODO-List as of January 31, 2017

1. look into 'standard-models' in ABM and check if and how they can be implemented using my terminology
2. try algebraic reasoning using my framework
3. Thorsten Meeting preparation
   - Meinen derzeitigen Stand (Paper und Code) aus Haskell Sicht
   - Idee für ein Paper: eine Spezifikationssprache für Agent-Based Modelling & Simulation, die quasi Haskell-Code ist - die Idee ist, der Frage nachgehen inwieweit der Spezifikation-Implementations Gap (mehr oder weniger) geschlossen werden kann, wenn wir das ganze in Haskell mit meinem Ansatz machen. Auch sind reasoning-Techniken hier sicher auch von Interesse.
   - Idee für ein weiteres Paper: meine derzeitige ABM/S Implementierung in Haskell mit FRP (Yampa) zu verbinden, was dem ganzen sicherlich ordentlich leverage gibt. Die Hypothese: die Richtung die ich eingeschlagen habe, würde sich sowieso nach FRP entwickeln, ausserdem erlaubt es wesentlich komplexere und größere Modelle zu implementieren.
   - FP Lunch Präsentation: steht immer noch aus, ist aber noch zu früh und das Thema war bisher nicht entsprechend, könnte aber mit dem ersten Paper eine passende Richtung vorgeben.
4. find interesting and large, complex ACE model simulating an economy
5. Bring PureAgents to Yampa
   - embed PureAgentsPar and Seq in Yampa: PureAgentsYampa so we can leverage the power of the EDSL, SFs, continuations,... of Yampa/Dunai.
   - implement agent monad: PureAgentsMonadic. but what is an Agent-Monad? build a monad to chain actions of the agent and always run inside an agent-monad
   - embed PureAgentsMonadic in Dunai
   - implement wait blocking for a message so far. utilize yampas event mechanism?
   - look into statistics of retries in STM `https://hackage.haskell.org/package/stm-stats-0.2.0.0/docs/Control-Concurrent-STM-Stats.html`: nothing revealing, all is running in parallel (although very inefficiently)
6. Push PureAgentsYampa by considering the next step: implementing an economics example at different levels of complexity e.g. Ionescus implementation of Gintis

7. Always ask the question: what is the difference to the OO approach?

# 4 Future-List as of January 31, 2017

1. Topics & Issues of Haskell Implementation

   - performance unacceptable: 1000 in haskell vs 100.000 in java is a shame on haskell, more should be possible. investigate using profiling both of CPU and memory: `http://keera.co.uk/blog/2014/10/15/from-60-fps-to-500/`. strictness & tail-recursion!
   - look into QuickCheck and HPC
   - problem: so far only agents with same static messagetypes, environment and states, can communicate: the agents are homogenous. how can we implement hetereogenous agents in this library?
   - implement a general-purpose rendering-frontend with gloss but let the simulation be driven by Yampa/SimulationBackend instead of frontend (not real-time)
   - implement PheroTrails: agents move on a 2d grid in a 8-neighbourhood and leave pheromone-trails which decay over time. the agents select the neighbourhood cell they move in the next step randomly according to the amount of pheromones present: the more pheromones are present the more likely they will move to that cell - note that this is relative: the pheromones of all neighbour cells are added up and normalized!

2. Simulation Model Ideas

   - IDEA: abm/s of a go game
   - IDEA: what about an ABM/S of karma and rebirth? add to genesis paper
   - IDEA: what about ABM/S generating sound? could be a perfect example for Yampa due to its signal functions. the sound is the result of interactions of agents which try to generate harmonies and agents trying to create dissonance
   - IDEA: what about abm/s creating drawings/art? 2d continuous and each agents path is drawn

# References


[1] Baker, H. K., Kiymaz, H., Alan, N. S., Bildik, R., and Schwartz, R. Market Microstructure in Emerging and Developed Markets. *Business Faculty Book Gallery* (Jan. 2013).

[2] Bowles, S., Edwards, R., and Roosevelt, F. *Understanding Capitalism: Competition, Command, and Change*, 3 edition ed. Oxford University Press, New York, Mar. 2005.

[3] Kirman, A. *Complex Economics: Individual and Collective Rationality.* Routledge, London ; New York, NY, July 2010.

[4] Spivak, D. I. *Category Theory for the Sciences*, 1 ed. Mit Press Ltd, Cambridge, Massachusetts, Nov. 2014.