Research Diary
PhD Studies

Jonathan Thaler

August 30, 2016

Started: 8th August 2016

Abstract

In this diary I try to keep track of the process of my PhD studies by recording my thoughts, my reflections, my (self) doubts and my philosophical ideas in connection with my research output.

2016 August 8th

First entry of research diary is a retrospective how I got to study a PhD at Nottingham. Due to the length of it I placed it in the appendices as Appendix B. Note that the next entry (10th) will give a short overview of the topic as it was at that point (10th).

2016 August 10th

Where do I stand now

On 21st June I had the opportunity to give a Phd-Seminar presenting my research ideas within 45mins + 15min discussion. This gave me the opportunity to make up my mind about the ideas I want to follow and which ones I want to reject. Also to better shape my ideas and to give a clear overview of them I started to write down a research proposal about 3 months ago. Of course this proposal is always changing and will continue to do so during the 1st semester as according to my supervisor the 1st semester is reserved to get started and to really carve out the research ideas and most important of all the research questions. So this is a still ongoing process but to give an overview where I stand NOW regarding the research direction I have will cite here the abstract of my research proposal which I have recently reworked (again) and which pretty well sums up the overall ideas and directions I want to follow:

Agent-Based Modeling and Simulation (ABM/S) is still a young discipline and the dominant approach to it is object oriented computation. This thesis goes into the opposite direction and asks how ABM/S can be mapped to and implemented using pure functional computation and what one gains from doing so. To the best knowledge of the author, so far no proper treatment of ABM/S in pure functional computation exists but only a few papers which only scratch the surface. The author argues that approaching ABM/S from a pure functional direction offers a wealth of new powerful tools and methods. The most obvious one is that when using pure functional computation (equational) resoning about the correctness and about total and partial correctness of the simulation becomes possible. The ultimate benefit is that Agda becomes applicable which is both a pure functional programming language and a proof assistant allowing both to compute the dynamics of the simulation and to look at meta-level properties of the simulation - termination, convergence, equilibria, domain specific properties - by constructing proofs utilizing computer aided verification.
To map ABM/S to pure functional computation the idea is to apply both Robin Milner's PI-calculus and category theory. The PI-calculus will be used for a formal modelling of the problem and allows already a basic form of algebraic reasoning. Then the agents and the process of the agent-simulation will be mapped to category theory because pure functional programming approaches complex problems from the direction of category theory in the form of monadic programming.
The application will be in the field of agent-based computational economics where the approach will be to take an established model/theory and then apply the above mentioned methods to it and to show that using them will lead to the same results.

Note that the goal is not to establish new economic theories but to provide methods and tools for deeper insight and verification in the context of agent-based computational economics - this is after all still a thesis purely rooted in computer science.

PhD Roadmap Semesters

1. Semester: Literature research & research questions. Topics:
   - Functional computation & programming: Agda, Haskell & monads, category theory, type-systems, computer aided formal verification
   - Formal ABM/S: pi-calculus, various formal agent-types
   - Finding application in agent-based computational economics: auction theory & auction types, market design

2. Semester: First publication: ? (Mapping auction types and ABM/S to category theory)
3. Semester: Second publication: ? (formalization of ABM/S and auction types in pi-calculus)
4. Semester: Simulation Framework, Finished Simulation framework implementation: basic framework with correctness & proofs
5. Semester: Third publication: ABM/S in Agda
6. Semester: Finalizing PhD - Writing final thesis combining all research & results.

Discussions with my Supervisors

- So far the proposed ideas are a huge amount for 3 years and I doubt it is realistic to do in 3 years. Also it is yet totally unclear whether it makes sense/is possible to map ZI-Agents and the auction-types to category theory AND write specifications in pi-calculus for both. Maybe only one of them is really necessary or maybe only part of each? This is to be discussed with my supervisors.
- Does it even make sense to study these auction-types with these types of agents? To look only at the dynamics? Is computational economics interested in these dynamics of the equilibrium processes? I really need to look into theoretical work on the various auction-types AND the computational economics approach to them.
- Do we really need the pi-calculus when we have category-theory and vice versa? Does not just one method suffice? Or is there a mapping between both methods / a connection?

August 2016 11th

Paper: Dependently-typed programming in scientific computing [?]

The authors give a short and nice introduction into the very basics of what economics is and what it tries to achieve: exchange of goods. They then go on and explain allocation and endowment and explain walrasian equilibrium and then use Agda for constructing fundamental economic proofs about Pareto Efficiency, Walrasian Equilibrium. Remarkable is their criticism of the equilibrium models:

> A general criticism of all these models is that they neglect the dynamical aspect of reaching the equilibrium situation. There is no known plausible mechanism which explains exactly how equilibrium

> prices can arise in practice. Walras' own proposal for such a price-formation mechanism involved an auctioneer. This is a central entity who can see all supply-demand imbalances and adjust prices accordingly, raising the prices of goods for which there is too great demand, and lowering those for which there is too little, in an iterative process. Even if one accepts that in some situations one could have an authority that might act as auctioneer, there is no general proof that the iterative process will eventually converge.

This is insofar remarkable as this is exactly the direction I want to head to and which was also the work of the Leverage-Cycle project in which I've written my Masterthesis: **compute (by simulation) and understand the dynamics of an equilibrium process**. One of the goals of this PhD thesis is to develop a tool in which such equilibrium processes can be modelled and simulated but it is still not clear in which economical context / which theory exactly - this is still an open question I have to deal with.

Another interesting point they make in the paper is that:

> The bad news is that most of these concepts are not constructive. Specifications of programs that take as input agents characterized by preference relations and initial endowments and return a Walrasian (or Nash, or correlated, ...) equilibrium can in general not be fulfilled. Even the so-called computable general equilibrium models are not, in fact, computable.

I can't yet tell the implication for my thesis but it will be interesting to see whether it has one or not.

Paper: Engineering Proof by Reflection in Agda [?]

The authors explain the reflection-mechanics of Agda and discuss how they can be put to use. In general reflection allows for a kind of meta-programming during both compile- and run-time by accessing and modifying the abstract syntax tree (AST) thus changing the structure and behaviour of the program during compile- and run-time. According to the authors, Proof by Reflection

> is the method of mechanically constructing a proof of a theorem by inspecting its shape.

Unfortunately this is paper comes too early as I don't know enough about Agda, intuitionistic logic, type theory and computer aided formal verification. I have to come back to this paper at a later point again.

Paper: Dependent Types at Work [?]

According to the authors

The aim of these notes is to give a gentle introduction to dependently typed programming for a reader who is familiar with ordinary functional programming, and who has basic knowledge of logic and type systems.

A few important topics I will going to clarify deeper after having worked through the paper and learned more about Agda and the theory behind it:

- *Dependent Types* where the type depends on another one e.g. a Vector of length n or a tuple (a,b) where a is less than b are dependent types.
- *The Curry-Howard correspondence* says basically that programming and proving will be the same.
- *Totality of Agda programs and Type-normalisation* Types must be normal otherwise the type-checking algorithm may not terminate
- *The connection of Dependent Types to the Curry-Howard correspondence* Still a VERY IMPORTANT open question.

August 2016 16th

Been working on [?] (Dependent Types at Work) which includes Exercises at the end of each subsections which are of great help to better understand the content if one solves them.

Category Theory vs. Pi-Calculus?

I've been thinking again about the combined application of category theory and pi-calculus in my thesis. Initially I thought that either of them is going to make it into the thesis because just one of them will be useful as a tool for formally expressing the type of agents I want to implement but I think this is wrong.
I now think that the pi-calculus will serve as a tool to formally specify the agents and their interactions in a process-calculus way and that I will use category theory for the implementation of the Domain-Specific Language which runs these pi-calculus specifications. It is interesting so see that the more I know and the more I learn the more I can picture and imagine and the clearer the road to my PhD gets.

Foundations of computation

Since I got in touch with the term of computability and computation I wondered what exactly is meant with it. I would now say that *computation is the manipulation of symbols according to specific rules.* More generally speaking the symbols are letters over an alphabet of a formal language and could be any e.g. the binary system where the symbols are just 0 and 1.

What intrigued me was why did the Old Gods (Turing, Church, Gödel) make such a fuss about computing natural numbers? I asked myself why in the world didn't they just use the binary system in which we can represent numbers?
The problem was my approach to computable numbers: the Old Gods never thought about the natural numbers in a binary system but in a purely mathematical sense defined by peano: the peano numbers. Those are inductively defined as being successors of the initial number zero. Thus 1 is successor of zero, 2 is the successor of the successor of zero and so on. Now they asked how one can perform basic arithmetic operations on such numbers? They asked how one can *compute* the addition, subtraction, division, multiplication of two peano numbers. This led to the invention of Gödel System T, Churchs Lambda Calculus and the Turing Machine. All of them had a different approach to computing the results but were shown to be of equal computational power.
While working on [?] (Dependent Types at Work) where the natural numbers were introduced in the style of peano it became clear for me that I did look at the whole problem from the wrong point of view. Now I better understand the approach of the Old Gods and with what they were struggling with. Still I have to think about how the problem of undeciability relates to these things.
What is clear now is that the computation over the natural peano numbers has the alphabet of zero, succ where the operations are implemented in the systems mentioned above: The Turing Machine performs the computations in imperative steps by changing a global (infinite) memory and Church and Gödel follow a functional mathematical style using so called primitive recursion.

August 2016 17th

As already mentioned in the entry of August 2016 11th, the authors of [?] tell bad news because "most of the concepts are not constructive". After thinking about this I have the feeling that this implies that I have to approach my agent-based simulation from a constructive direction: both the agents and the simulation itself have to be *constructive*. It is still not very clear to me what that means but I guess that one can draw parallels to the peano numbers: I think the peano numbers are an inductive *constructive* way of, well... *constructing* the natural numbers and not just postulating their existence - each number has to be computed! We will see whether my intuition is right and how this will apply to ABM/S. After having read the entry on Constructivism (Mathematics) on

Wiki it is now a bit clearer. To quote Wiki "Thus the proof of the existence of a mathematical object is tied to the possibility of its construction.". This is exactly what happens with the pano numbers: they are constructed and thus they exist - they are computable!
I am now in Section 3 of [?] (Dependent Types at Work) and have decided to

first read through some more introductions to Agda and dependent types before continuing as I first have to understand more basic things about this theory.

Found a very nice lecture about *Interactive Theorem Proving for Agda Users* at

`http://www.cs.swan.ac.uk/~csetzer/lectures/intertheo/07/interactiveTheoremProvingForAgdaUse`
`html`.

August 2016 22nd

I am currently on holidays with my lovely girlfriend Sarah. We started on the 20th towards Denmark and are now in Sweden (camping at Stocken) and I will return on the 7th of September. Due to lots of free time I brought a bit of things to read:

1. Paper by Per-Martin Löf
2. Yellow Paper of Ethereum (TODO: cite)
3. White Paper of Ethereum (TODO: cite)
4. Handbook of Market Design (TODO: cite)

0.1 Ethereum

Based upon a hint by a good friend on (2016 August 17th) I started exploring the idea behind Ethereum which immediately fascinated me. This is a topic I will definitely pursue as a private interest during my PhD Studies. I hope to be able to participate in a project or have the idea for one myself and maybe it can be combined with my PhD Studies (agents running in Ethereum?).
After having read the White Paper (TODO: cite) and the Yellow Paper (TODO: cite) of Ethereum, I am even more determined to participate in this huge new thing of the internet. I am very excited to get my hands on it as I feel its a very interesting playground allowing for a huge variety of opportunities both in learning and earning money.

To sum up what fascinates me is that Ethereum

> ... provide a massive boost to other peer-to-peer protocols by adding for the first time an economic layer.

This allows

> ... anyone to build what is essentially a command line application run on a virtual machine that is executed by consensus across the entire network, allowing it to modify a globally accessible state as its "hard drive".

This immediately spawned the following ideas in my mind:

1. Implement an Ethereum client in Haskell. This is an amazing way to improve my Haskell and finally apply it in a real-world Application which touches on a huge range of computer-science topics. I've seen that attempts to implement an Ethereum client in Haskell have been already made, I will look at them when I return from my holidays but I want to attempt it nonetheless as it will hugely improve my Haskell skills. It will be a huge challenge because its a very complicated protocol including the implementation of a Virtual Machine (Ethereum Virtual Machine, EVM) but fortunately everything is described formally and very precise in the Yellow Paper (TODO: cite). Things to implement:
    - Ethereum Virtual Machine
    - Proof-Of-Work Hashing Algorithm - should be done on GPU otherwise too slow
    - Communication with Network
    - ?
2. Learn and play around with Serpent the programming language of the EVM to implement contracts. I want to try to implement various types of auctions (batch, continuous, simultaneous ascending, sealed second price,...) and voting-systems.
3. Set up a private account in the public Ethereum Network.
4. Set up a public Node for participating in Ethereum and mining Ether.
5. Thinking about an interesting DAO (Decentralized Autonomous Organization) to implement in Ethereum

Also Ethereum could be the perfect Use-Case for applying my ABM/S to computational Economics:

- Simulate Ethereum: dynamics of ether or gas? Ethereum is a very itnersting agent-based system (peer-2-peer with an ecoinomic layer
- Simulating dynamics of DAOs in Etherum?

August 2016 23rd

After meditation and after looking through "Handbook of Marketdesign" (TODO: cite) it struck me: Why don't do I just use the model and simulation of my Masterthesis instead of keep looking for an awesome model which I still coulnd't find? The vorteile are

- I am already very familiar with the model, simulation and results.
- I am already familiar with the theory behind it by Geanakoplos (TODO: cite Leverage-Cycle by Genakoplos)

- It is complicated but simple enough to serve as a starting point to apply and proof my methods.
- It includes a few interesting aspects which can be proofed using my methods e.g. under which circumstances does trading stop?
- It can then be extended to a full-fledged framework as originally intended.

TODOs

Starting from now on I will include TODO-subsections which include well... things to do so that I have a small roadmap ahead. When a bullet-point in the TODOs is done then I will write an entry with the updated TODO list.

- Read the following chapters of "Handbook of Market-Design" in the given order: 2,3,12,10,13,16,17
- Read my Masterthesis again.
- Rework my Reseach-Proposal and include the approach of my masterthesis and think about how I can apply the formal methods to it - this will then be the basics of the initial meeting with my supervisors unless I will find a kick-ass idea regarding ABM/S of Ethereum
- Get basic understanding in Agda
  - Work through "Dependent Types at Work"
  - Look at Thorsten Altenkirchs Lecture "Computer Aided Formal Verification
  - Read online Lectures about Dependent Types: TODO
- Haskell Learning: Implement the core of my Masterthesis in Haskell as I know it at the moment (no Monads): the core is the replicated auction-mechanism where every Agent knows all others.
- Haskell Learning: Start implementing Ethereum Client.
- Alternative Topic of my PhD: ABM/S of Ethereum as a decentralized marketplace. Maybe these questions help a little bit (TODO: explain, see Handbook of Market-Design Introduction to Chapter 1) but unfortunately I know not enough about it but hope that I can get deeper into it soon.
  - How well does it provide thickness?
  - How does it deal with congestion?
  - How does it make the market safe and simple to participate in?

August 2016 24th

Yesterday night I had some (self-)doubts about the direction of my PhD, more specific the application of ABM/S to agent-based computational economics.
Doubts came up that I can properly understand and get into this field but after thinking about it a while, it became clear that ABM/S is - unless one does fundamental theoretical work of ABM/S itself - always applied in the context of another domain e.g. Physics, Computer Science, Economics,... I plan to do fundamental theoretical work in ABM/S itself (formal methods) but I will apply it in the field of agent-basec computational economics thus I definitely have to learn the basics of this field and become an expert in the specific subfield I plan to work in (equilibrium in trading-processes).
I chose the field of agent-based computational economics as I have already a bit of experience in it due to my Masterthesis and because I think it is a fascinating field and is of very importance to Economists to better understand dynamics of static postulates (e.g. equilibria).
The main question is, what exactly I want to research in this field of agent-based computational economics (ABCE)? The answer

> There exist Equilibrium Theorems in trading but no description of processes reaching them. I want to look at simulating such processes which may or may reach some steady state which may or may not be an equilibrium.

Also while thinking about this fact I came up with the thought:

> The Economists focus completely on their equilibria. Maybe this is the wrong way to go and we should abandon to try verzweifelt reaching equilibria but should only look at the dynamics of the processes and how they unfold - I believe the dynamics are of most importance as they really affect the world and not equilibria.

Nonetheless to abandon equilibria in favour of dynamic processes I must start with the very basics of equilibria to understand them, to follow the road of the economists why they are so obsessed with them and to know to talk about them and to encounter economists in discussions - I REALLY have to know the equilibrium theory (to be found in Microeconomics, which I should study in a basic form like the book I own: TODO cite). Also I should really know Market Microstructure because it focuses more on the dynamics of prices and of trading and I already got a nice book for this (TODO: cite)

Thus the plan is to start from my masterthesis. Take the model and simulation (continuous double-auction with Zero-Intelligence Traders) as they are without any change and develop and apply the methods to it and also do formal verification and proofing of various properties like convergence (termination) and deadlock free trading and buyer optimism ¿= seller optimism. The used methods will be as already mentioned Category Theory, Pi-Calculus, Haskell and Agda.

Then after having developed the methods and seen them in action then I can work towards a more general application of them to research and study the dynamics of trading-processes and abandoning the focus on equilibria. Until then I have to have studied severely ABCE, equilibrium and auction theory, have an overall basic understanding of Microeconomics, have a very good understanding of Market-Microstructure (the key properties e.g. volatility,...) and have studied basics of Market-Design.

I know that this seems extremely ambitious but I think it is a very good point from which to start my PhD adventure - we will see what the outcome will be in the end.

TODOs

I think I should abandon the idea to do something with Ethereum as a decrentralized marketplace in my PhD, thus abandoning it from my TODOs. Also I think I should attempt to use Monads in reimplementing my Masterthesis.
Thus the updated todos are:

- Read my Masterthesis again.
- Rework my Reseach-Proposal and include the approach of my masterthesis and think about how I can apply the formal methods to it - this will then be the basics of the initial meeting with my supervisors. Also include the thoughts of the entry to the diary of 2016 August 24th.
- Read the following chapters of "Handbook of Market-Design" in the given order: 2,3,12,10,13,16,17
- Get basic understanding in Agda
  - Work through "Dependent Types at Work"
  - Look at Thorsten Altenkirchs Lecture "Computer Aided Formal Verification
  - Read online Lectures about Dependent Types: TODO
- Haskell Learning: Implement the core of my Masterthesis in Haskell as I know it at the moment and trying to incorporate Monads: the core is the replicated auction-mechanism where every Agent knows all others.
- Haskell Learning: Start implementing Ethereum Client.

August 2016 30th

Having enough time in my holidays to think and reflect about the direction of the computational economics part of my PhD I seem to go in circles but I think

I finally agreed with myself about the right approach.

The economic theory & model behind my masterthesis are too complex to start with. The definitive approach is now to start with the classical paper of Gode & Sunder (in which they introduced zero-intelligence (ZI) traders within continuous double-auctions) to develop the basic methods and tools on an already researched topic and then to gradually broaden to a more general framework (CDA variants) with focus on dynamics instead of equilibrium.

TODOs

- Read Gode & Sunders paper again (TODO: cite).
- Read Everything what you wanted to know about continuous double-auction paper again (TODO: cite).
- Read my Masterthesis again.
- Read the following chapters of "Handbook of Market-Design" in the given order: 2,3,12,10,13,16,17
- Rework my Reseach-Proposal and reformulate it to cover the above idea. This will then be the basics of the initial meeting with my supervisors. Also include the thoughts of the entry to the diary of 2016 August 24th. And include essence of entry to diary of 2016 August 30th.
- Get basic understanding in Agda
  - Work through "Dependent Types at Work"
  - Look at Thorsten Altenkirchs Lecture "Computer Aided Formal Verification
  - Read online Lectures about Dependent Types: TODO
- Haskell Learning: Implement the core of my Masterthesis in Haskell as I know it at the moment and trying to incorporate Monads: the core is the replicated auction-mechanism where every Agent knows all others.
- Haskell Learning: Start implementing Ethereum Client.

I also had a contemplation with Samuel about the direction of my PhD and he told me that "Never stop following your feelings and intuition. If you follow that road it cant be wrong." - thus I am on the right path as I always follow the feelings and the intuition which comes up during reflection on the topics.

A The evolution of my computer science interests

This is a retrospective of how my interests in computer science evolved into the current state.

I was and am no genius. I was also not extremely smart, sometimes I was slow in understanding, I was good enough but what set me apart was: if I understood something, I've understood fundamental and I could transfer it to other areas as well. Also if I really wanted to do something I pursued it so long until I could do it (gameengine). And the most important
There was always a bit of playfulness in my approaches: it was the mere curiosity how things would unfold, the fun in exploring unknown territory - when I had the feeling I understood enough I moved on to new territories. Thus the major shift of interests from starting programming with age of 15 until now with age of 33 from basic text games to GUI programming to 3D graphics programming with OpenGL 4.0 to web 2.0 development to enterprise back-end applications to operating systems to embedded systems to system simulation and now to the very basics and mysteries of computation itself.

B How to get a PhD-study at Nottingham

The idea to do a PhD at Nottingham came up during the visit of Nottingham for the Leverage-Cycle Project which was 21st - 23rd May 2015. I was part of a small team consisting of Thomas Breuer, Martin Summer, Hans-Joachim Vollbrecht and me and our goal was to demonstrate Simon Gächter our software for running a specific type of economic experiment with real agents. After 3 sessions we had him convinced and he agreed to join in a research-cooperation in which the goal was to get a grant to further study the equilibrium theory of Geanokoplos. The idea was that in the funding of the research-project a doctoral study position should be included which would be taylored towards me so I could come to Nottingham and do a PhD in computer-science but in the context of the Leverage-Cycle Project.
The problem of getting such research-project grants is that it is a quite long process and the outcome is very unsure and it is highly probable that one gets rejected. Thus when starting the preparations for a PhD study at Nottingham in October I had to search for a 'Plan B' covering the case of the rejection of the research-grant. Also I tried to make use of our connections to Nottingham so I first contacted Martin Summer who gave me contact details of Uwe Aickelin, the Head of the Computer School which has his research-interest in Agent-Based Modeling/Simulation and Data-Mining. I wrote him an e-mail about a potential supervision and he quickly replied tell me that he will leave Nottingham and move to the chinese campus of the University of Nottingham but referring me to Peer-Olaf Siebers who is also active in the ABM/S field.
Peer-Olaf quickly replied and we did an initial Skype Call to get to know each other. We both then agreed that we would like to work with each other, especially my interest to go in a (computational) economics direction was very well received by him as he tries to get foot into this area as well but hasn't managed to do so far.
Now I had to think about what I would like to do in my PhD as so far I had only a few vague "Visions" and brainstormed a few vague Ideas. The overall direction was clear: I wanted to do something in the field of ABM/S and economics but all approached from the direction of computer science as this is my very field of expertise and I would get rejected by economists because of complete different wording and approach. Prior to my application I already had the idea to head into the direction of functional programming in the field of ABM/S as the dominant method is Object-Orientation (OO) so I thought it would be a nice idea to try something very new.
The idea I came up with was to apply functional programming in Erlang to ABM/S to simulate "something in economics" where something was "trading in virtual economies", "High-frequency trading", "Electronic Trading Platforms"... which could be boiled down to "Market Design". It was an intense and interesting process as I had to read quite a few papers and had to "clean up" and clarify my ideas and my mind. So after about 2 weeks of intense reading, reflecting and meditating I decided to go into the direction of "applying functional programming with special regard of ERLANG to ABM/S with a case study

in simulating the influence of different auction- and market types in electronic trading platforms."
I presented this to Peer-Olaf in a skype-call on the 4th of january. He told me that he will get a co-supervisor because he is no specialist in functional programming and more important, that I should apply for one of the 10 studentships granted by the school of computer-science. He made it clear that he will accept me as a student only if I will come to Nottingham and only if get this Studentship.
So the next goal was to apply for the studentship which was also a quite time-consuming process as you have to REALLY nail it - it's gotta be perfect otherwise one gets rejected really fast as you are competing with lots of other applicants. I had to provide a document containing the following:

1. Half a page describing reasons for wishing to pursue a PhD at Nottingham.
2. Half a page describing the proposed research area and topic (including a few references).
3. A demonstration of my technical writing skills: my Masterthesis.
4. Contact details of a supervisor at Nottingham who has already agreed to supervise my PhD.
5. Contact details of 2 Academic Referees - I did include Thomas Breuer and Simon Gächter
6. A CV including my degree classes.
7. A scan of my Bachelors and Masters degrees of all courses and the diploma supplement of both.

The studentship would start on 1st October 2016 and would grant me 14,507 pounds a year AND tuition fees (which are about 9,000 pounds a year). I had time until the 29th of February but Peer-Olaf told me not to wait until the deadline but to submit maybe one month earlier to be able to react if something is rejected. With a little help of my Prof. Hans-Vollbrecht, Thomas Breuer and BIG help of Peer-Olaf I did write and compile all the necessary information and applied for the Studentship on the 18th January. Peer-Olaf also did find a second supervisor - Thorsten Altenkirch from the Functional Programming Group who has his interst in Computer Aided Formal Verification and Type Theory, which makes a perfect match.

After about 2 months on the 16th March I got news from the school of computer-science that I have been allocated on of the school PhD studentships. I nearly freaked out so happy was I but deep within me I always had the feeling that things would work out and that I'd go to Nottingham - the one way or the other.

B.1 Studentship received - what now?

Getting the studentship is one thing but organizing everything is another big thing. What had to be done was:

1. Officially apply with Admissions Office so a conditional offer can be made. Conditional means that a few things need to be provided in order to receive an unconditional offer which finally guarantees me that I will study there. The things to provide were an IELTS test (see below) and the referee's reports (see below).

2. Apply for the VC's Scholarship for Research Excellence (EU). The school of computer-science will guarantee a fully funded place but wants me to apply for this scholarship. If I receive it (which I did) then they just have to top up to 14.507.

3. Provide an Academic IELTS certificate with overall score not less than 6.5 and in no part less than 6.0. I easily passed getting an overall Band Score of 8.5 / CEFR Level of C2 with having achieved Listening 9.0, Reading 8.5, Writing 7.5, Speaking 8.0.

4. Provide a referee's report of both my academic referees (which they were happy to do).

5. Apply for an accomodation in one of the many student hostels on the campus. I opted for Melton Hall, non self-catered, En Suite (including a bath in my room) for 5,890.5 Pounds a Year.

6. And most important: prepare myself and refine my topic so that I come well-prepared to Nottingham.

I could provide all the details and got an unconditional offer which made me a future student starting my PhD Programm on 1st October 2016. But now I had to really focus and dig into the topic

B.2 Preparing for my PhD

Since January I did a lot of preparation and learning for my PhD (which was done parallel to 24hrs/Week Job)

- Learned Erlang – implemented basic version of masterthesis CDA

- Learned Haskell (Graham Hutton Book & Tutorials on wiki.haskell.org on Monadic programming) – also implemented basic version of masterthesis CDA

- Gained basic understanding of what Agda is and what is is capable of (Wikipedia of agda, Thorsten's computer aided formal reasoning lecture, learnyouanagda, ...)

- Reading and studying book ”Category theory for the sciences”
- Reading book Actor Model by Agha Gul
- Reading book PI-calculus by Robin Millner's
- Reading papers on formal ABM/S
- Reading papers on PI-calculus
- Reading papers on ABS in computational economics
- Reading papers on market designs

Initially I focused too much on the method but couldn't really argue why I was using it and what I want to achieve in the end. Writing a Research Proposal, giving a Presentation of my PhD Topic at FHV and discussing my ideas with others was of a great help to focus more on these missing points.

The insights during the preparation-process were the following:

- Erlang is a dead-end as it is not a pure functional language and reasoning is not really possible due to parallel nature and language constructs. Haskell is the way to go.
- The Actor Model will most probably not be the direction I will head to because research seemed to move into different direction than that was I need, my agents are most probably of different nature. Actor Model is more about the concurrent and parallel interaction between agents (actors) which is an interesting aspect but most probably not the focus of my research (and would open up a whole new pandoras-box).
- I want too much, I need to focus on less - this is a big thing to discuss with both my supervisors and I am sure they can help me with narrowing down my research topic.