Teaching Statement

Klaus v. Gleissenthall

Teaching Philosophy To me, teaching is ultimately not about getting students to learn and apply some particular language or algorithm or mathematical technique; it is about inspiring them to learn more about the subject and enabling them to see the beauty in the reasoning principles that underpin it. One of the first courses that really got me interested in pursuing computer science research was a seminar on machine learning algorithms that were able to compose music. I was drawn in by the intriguing idea of machines being able to achieve such an innately human and creative task as composition. What ultimately captivated me, though, was the beauty and simplicity of the underlying mathematics which enabled it.

At the core of this beauty is always clarity: in science, no definition or algorithm has to be accepted as a given, in its own right, but it must always earn its place. "Why" is always a good question!

One of the things I find most rewarding is to help students dissect a problem until all its part fit together and make sense. The way to achieve this is often through examples. I always aim to draw these examples from real world problems that students care about. Finally, I think that the best way to understand a method is by implementing it.

I find that teaching also presents a great opportunity to make students think about program correctness not as an afterthought but as a design principle. This is closely tied to my research interests: the philosophy behind pretend synchrony is that one can often structure algorithms in a way to make reasoning about their correctness easy. This is not only helpful when formally verifying their correctness but even when manually reasoning about them. I aim to incorporate these principles into my teaching.

Teaching Experience I have helped to teach two verification ("model checking") courses at Technische Universität München and tutored a Prolog course at the University of Cambridge. The verification course involved giving lectures, while the Prolog course consisted of giving tutorials to small groups of students. I enjoyed these experiences and think that teaching is one of the essential ways we can have impact as researchers: by passing on our way of thinking.

Mentoring I have advised several PhD and undergraduate students who worked with me during my time as a post-doc at UCSD. My style of mentoring is collaborative: my main goal is to realize interesting projects and to do cutting-edge research together – while teaching some important technical and research skills along the way. An essential ingredient for this is the friendly and positive environment that results from working on a common goal. I aim to strike a balance between helping when help is needed and giving enough autonomy. A quote that got stuck in my head sums up this philosophy nicely: "when they're stuck writing the code, I commit the first line; when they're stuck writing the proof, I write the first lemma".

Future Teaching My training has put me in a position to teach undergraduate courses on programming languages, logic, formal methods, compilers, software engineering and introductory programming. I would also be happy to teach undergraduate classes outside my immediate areas of interest. I am excited to teach seminars and courses at graduate level. A natural topic for me would be the verification of distributed systems and hardware security. I would also enjoy teaching courses on related topics such as verification, logic, the internals of SAT and SMT solvers, and concurrency theory.