Good day everyone,

As some of you may not know, my name is Thomas Apeltauer and I come from the Czech Technical University in Prague. I stand here today to present you my doctorate research. The topic of my research is called “*Automated testing of models of cyber-physical systems*”, so I am working in a field of formal verification, digital design and computer science.

You might wonder, what is the Czech guy doing here all the time? Well that is certainly a proper question, let me clarify it. On this illustration you can see a vision of a future. A household full of “*Internet of Things*” devices, all of them connected to each other sharing information, data and states. This way we are building an “*intelligent network*” capable of automation many of our daily routine. The same approach is used in the industry and it is called an “*Industry 4.0*”. Engineers usually use MATLAB and Simulink to create such complex systems as this software has proven to be very usefull. All these visions assume that we will be able to create reliable and safe cyber-physical systems. Verification of such systems is exactly the area of our research.

So, what are exactly cyber-physical systems? There are many definitions out there in the academic world, but the base idea is that cyber-physical system consists of some computational unit, sensors and actuators which interact and measure real physical world. The requirements on such CPS are usually extremely high. Automatic transmission plays a crucial role in the vehicle functionality and its malfunction could be lethal to the passengers. To verify that our automatic transmission is not going to kill our customers, we must verify the system itself, mainly it’s design. We define requirements on the design and we verify that system meets them. Such requirement could be for example statement “Vehicle speed will not exceed 120 km/h” or statement “The engine speed will not reach 4500 rpm”. To verify these statements, we are trying to find an input throttle schedule such as on the picture here.

Because creation of safe, functional and reliable cyber-physical systems is not a trivial task, a V-design cycle was created to help engineers achieve it. This cycle covers all parts of the product development phases and helps to define what should engineers and designers do in each step. I will not go into much detail now, but I would like to emphasize this step. During this phase engineers are usually creating models of the desired system. Usually in MATLAB + Simulink + Stateflow. Models help them to spot design flaws and they are essential to the verification, because they can be simulated.

So how does it look like now in the field of automated verification? If you are an engineer creating new cyber-physical system and you have set of requirements together with models of the system components, or the whole system, you can use modern solvers to verify your design automatically. These solvers (here represented as an algorithm icon) use different optimization algorithms, be it Simulated Annealing, Genetic Algorithms, Cross Entropy or Uniform Random Sampling. General approach in the field is that you will provide some sort of specification of the requirements on the system, you will provide a model for simulation and the solver automatically runs hundreds to thousands of simulations to find a witness and thus falsifying given properties and proving the design is wrong. Sadly, we do not have enough computational power to run all the possible simulations and thus verify the design, so if a falsification witness is not found, we are still not sure about the correctness of the design. The interesting thing about this approach is that all solver techniques are considering the model as a black box and does not take into an account a structure of the model. This approach is good, but it is not good enough.

We can still improve it by opening the black box (meaning the model) and taking into consideration the inner parts of it. I will not go into much detail here, but let’s just say that there are many possibilities how to guide the solver algorithms to the parts of the state space where it is more likely to falsify the requirements. This is exactly what our research focuses on and what I am currently working on.

Now this was first half of my presentation which should give you the basic overview of our research and the area of the automated testing of models of cyber-physical systems. Now I would like to talk a little bit about the schedule of our research, about concrete parts of our research in a little bit more detail. The ultimate goal for me is to write a conference article. I should have it by the end of the 1st half of this year. In order to write a conference article, I have to go through these phases:

1. I need to learn and understand Metric Temporal Logic (MTL) specification, because it is heavily used for specifying requirements on the real-time systems if we want to verify them
2. I must understand benchmark models used for development of actual tools for verification
3. I have to design experiments, using already existing tools for verification, run them and measure them. The purpose of the experiments will be to find a weakness in existing tools -> An opportunity for us to create a better solution
4. During experiment phase I will be simultaneously collecting results and then evaluating them

When I finish all these steps, I will have enough material to write an article and possibly publish it on a proper conference.

First step on this journey was to understand the MTL and learn how to use it in practice. It was defined by Ron Koymans in 1990. Back then there was no tool for specifying requirements for real-time systems. Classical temporal logic was insufficient, because it offered only qualitative operators, but engineers needed also quantitative operators. This version of temporal logic has a certain metric point structure, which you can see on the picture here and which is nonarchimedean. It also provides us with metric versions of classical logic operators. G means that formula holds always in a future, P means that formula is true at least once in the future, H means that formula has always been true in the past and P means that formula was true at least once in the past. Operators until and since are quite self-explanatory. Apart from that we are also able to specify time range for all operators as we will see on next slide.

Here we have some examples of usage of MTL in practice. Typical promptness requirement can be express as is shown on the first line. Requirements from the example with the automatic transmission can be expressed by one MTL formula as shown above the picture. Last but not least the definition of a time-out on an event e is a very useful thing.

There are of course many benchmark models which can be used to measure the performance of verification tools. I have shown you the automatic transmission benchmark model, there is also one other popular benchmark and that is the “*Room heating benchmark*”. This model represents a system with 10 different rooms, 4 heaters and a controller ensuring, that temperature in the rooms will not fall under 14 degrees. You can see one run of the simulation of this benchmark on this picture.

Now having benchmark models is nice, but we need a solver tool to put them into use. One example of such solver can be S-TaLiRo tools, developed by a research team at the Arizona State University. This tool comes in a form of a MATLAB toolbox and it can be used to verify models of cyber-physical systems against an MTL specification. It literally searches for counterexamples by minimizing so called “Robustness Metric”. It performs hundreds of simulations over the model, based on the complexity of the model and performance of the machine.

To sum up our research let’s answer few basic questions. What is our goal exactly? Well we want to create new technique of using the inner structures of models of cyber-physical systems to enhance its verification process. How do we mean to achieve it? We will take for example the S-TaLiRo tool and measure its performance over a lot of other models. This way we will identify weak spots and we will be able to suggest a better way. If we will succeed in a creation of such new technique, we will generalize it and form a new innovative verification method. And what are we working on right now? Currently I am preparing experiments with S-TaLiRo tool. I am creating running scripts for new models and I am about to measure the S-TaLiRo performance on these models. For example, model of PID controller with 2 degrees of freedom for DC motor, or LineFollower project, which is practically a model of a robotic vacuum cleaner, or an Electric Vehicle model and so on.

You might wonder how does a script for S-TaLiRo looks like? This picture describes it. Basically there are parts specifying the model to be loaded, the MTL specification to be verified, initial conditions, range of input signals and also the time of simulation. This is a simple script for the “*Room heating benchmark*”. For more complex models there are naturally more complex scripts.

That is all from me for today. Sadly, I have only scratched the surface of this topic as explaining all details would be quite time consuming and I guess also boring. Thank you very much for your attention and if you have any questions, please don’t hesitate to ask.