Transcription of the audio interview with Benjamin Lion

My name is Benjamin Lion and I work at the Inria center at the University of Rennes. I study program side effects. These are effects which are not functional, not in logic, but which are consequences of the execution of a program on memory or on its environment. One of these side effects is time. A program takes time to run.

I'm interested in methods for understanding this time that runs, and in areas where mastering time is critical. And one of these areas, for example, is, let's say, the Internet of Things, where you have a controller that interacts with its environment, having instructions that execute at a precise time. This is crucial.

Today, we can't prove it formally. We have analysis methods, but in a generic way. When a program preserves time properties during compilation, it's difficult to have a generic method. So there are dysfunctions caused by the fact that we can't understand precisely how time evolves during execution. But there's also an ambiguity for the developer when he programs to know a priori what the behavior of his program will be, and so for the moment, there's no way of having security over the execution of his program, a priori.

I've been awarded a Marie Curie post-doctoral fellowship, which will enable me to spend two years looking into ways of certifying the time taken to run the program. So, there are two main challenges. The first is to be able to express time in semantics, which would enable us to study the semantics of a program by including temporal aspects and to deduce time properties of a program in relation to time properties of sub-parts.

The second objective is to preserve these properties through a compilation chain, which would make it possible to prove properties on a program and have these properties maintained because the program is executed. I'm not going to create the compiler from scratch, but I'm going to base myself on work that has already been developed, consequently; at Inria, which is the certification of a compiler for the C language. This compiler has been formalized in a proof assistant and several properties have been proven, including semantic preservation. In other words, the program we run after compilation has the same behavior as the program we wrote in C.

The certified compiler is called CompCert and the proof assistant is called Coq. And that's a big area of research and also a big area of application. All applications that are time-sensitive would need certification to ensure that their program runs in a precise time. So systems like connected industry, where programs run at precise times to operate on a machine, or more critical systems that have to interact with physics, need to include time property preservation certifications in their compilation.