Supporting Formal Method Teaching with Real-Life Protocols

Hugo Brakman Vincent Driessen Joseph Kavuma Laura Nij Bijvank
Sander Vermolen [*]
Radboud University Nijmegen

May 2006

Abstract

In the world of computer science, formal methods play a primary role in the development of student minds and ability of abstract problem solving. Formal methods form the foremost technique that enables students to be trained in breaking complex questions up into abstract, manageable, pieces and to solve them using models that they themselves constructed. While the advantages of formal methods seem clear, students *en masse* tend not to attend the courses. This short paper analyses the core problems underlying this phenomenon and shows by example why the practical assignment that has been carried out by the five of us during a course in embedded systems, has contributed to the goal of formal method teaching.

1 Educational aspects

There is considerable evidence that the teaching of formal methods contributes largely to the student's understanding of problems, abstract reasoning skills, and their ability of elegant problem solving. However, these skills are not taught at once, but are part of a large educational programme, much like how reasoning skills are taught in high school. The consequence is that the yield of this kind of teaching is settled in the long term.

Students on the other hand, tend to have a more narrow focus, valuing their study progress by looking at short term efforts and results. Important roles in their course selection are the so called "fun-factor" and the amount of (applicable) skills or knowledge that will be gained by following the course. As a typical example, a student would preferrably follow a web programming course than a course in formal methods.

Among the problems that formal method teaching faces currently, we may distinguish the following:

"What's the use?" Students simply do not see the practical use of "all this math". This is the main problem for formal method teachers to face. This problem incorporates a few consequences.

First and foremost, motivation for students drops significantly if they must do things when they have no clue about what they will gain from doing it. Telling them that they will learn "abstract reasoning skills" will not work either, because that sounds much too vague. There is a big necessity for rolling out a clear roadmap with targets in the student's curriculum and sticking with that map in order to keep them motivated.

Furthermore, there is competition from other fields of science. Especially nowadays, with the computer and IT-business luring people from universities, students foresee a better future by pursuing a practical master degree. If universities do not succeed in waking up the interest in formal methods, computer science may start getting loose from its (mathematical) roots.

Lack of visualization Considering the problems *within* the formal method courses, lack of visualization is the first one. Teaching plain dry mathematics is not only dull, it also is not the most effective route to the mind of the students. The human brain learns through association. Cognitive research has already shown that education through visualization enables a better understanding of matter, because mental models are constructed intuitively and more solid.

This would also explain the apparent resistance to these courses. An aura of complexity, mathematical sophistication and unfamiliarity surrounds formal methods courses. This aura is kept alive by the thought that insights are something that someone has from birth, and cannot be learned to a high degree. However, insights can be *enabled*, not only gained. Repetitively letting students "solve equations" narrows their focus and does not connect to the goal of what they are doing it for. Illustrative material will help their brains to enable insights through visualization, much like how a picture says more than a thousand words.

Application of knowledge The last problem is a generally known one, which occurs at the moment the students have gained the desired knowledge on formal methods. Since generalization is one of the most important focusses of scientific teaching, the connection to real world problems might be lost. A dedicated task of formal method

[*] Contact author. Email sandervermolen@student.ru.nl

teaching should be the establishment of a link between theory and practice, not the practice itself.

When enabling insights, teachers must beware that these insights are not isolated in the minds of their students, but are made accessible. One way of doing this, is to make sure that visualized concepts are being linked to situations where that type of knowledge is applicable, if possible. This effort would especially avoid freshmen developing an anxiety for mathematics and would also contribute to help students see the use of formal methods.

2 Solution

The solution outline we propose and which we can recommend through experience is the following:

Take a real-life problem and a known solution to this, or a sub solution of it. Examples are known network protocols such as Ethernet or as in our case Bluetooth. Let the students make an abstract model of this, that can be used as input to a known model checker. The next obvious step is then to let them use this model checker to verify properties of the model, that together prove the correctness of the protocol.

We will now look into our project more specifically, which will follow the outline presented above.

The purpose of our project was to do a formal verification of part of the Bluetooth protocol. This wireless network protocol consists of several relatively separate phases, which gave us the possibility to examine part of it and still be able to draw very useful conclusions. We have examined the inquiry part. This is usually the first phase in a Bluetooth communication. The purpose of this phase is to discover other devices and find out some basic facts about them.

How we would tackle the problem was mainly up to us. We have decided to alternate individual study with discussions of the problems we had encountered. In case there was a problem we were not able to solve ourselves, we could always ask for assistance. But in practice, because of the setup we have used, we appeared to be able to solve most issues ourselves.

We have split up the content of the project in three parts: investigation of the protocol, modeling the protocol and finally using our model to prove several theorems about Bluetooth communication.

Investigating a protocol description does not appear to be something one does for fun. At first hand, we shared this opinion. However the crucial issue appeared to be in how to investigate the protocol. Doing a broad and superficial survey can be hard and will usually result in forgetting it some days later. In contrast to this, we have investigated part of the protocol in very much detail and made a very narrow survey. This did not only limit the number of pages we had to read, but it also gave us the possibility to understand and check most of the details of the protocol.

We had divided the Inquiry phase among the five of us, discussing all parts one of us might not have understood. Obviously, this resulted in a good understanding of the protocol, but it also resulted in finding some unclarities and some possible inconsistencies in the protocol document. Which makes investigating it quite satisfactory.

Having gained enough knowledge of the protocol, we could move on to the next phase of our project: modeling Bluetooth. The model checker of our choice was UPPAAL. UPPAAL was able to give us a good interface and a useful model checker. But in contrast to many of the other tools we have seen so far, UPPAAL also provided us with a very useful simulator in which one can view and understand traces and use these to further correct the model, or to draw conclusions.

Now, understanding the protocol was one thing, modeling, an entire other. Some of the problems we had heard in theory appeared to be obstructing the design of our model more than we had expected. And especially many of the idealistically stated solutions appeared to have some unforeseen drawbacks. Nevertheless, given some time, we were able to find the solutions to most of these problems, learning many ins and outs of the used theories on our way. Mainly the fact that we have solved the issues ourselves gave us great insights in what the 'idealistically stated solutions' were really about.

The last part of the project consisted of using our model to verify correctness properties of the Bluetooth protocol. Unfortunately this took UPPAAL a lot more time than we had accounted for. The only solution was to abstract our model from some of the hardware details, to make its complexity suitable enough to do verifications of some of the properties we had come up with. But the difficulty was to keep it realistic enough to still be able to draw conclusions about Bluetooth in real-life. Our effort in trying to reduce the complexity eventually resulted in a model that was only slightly simplified, but well suitable to verify our properties. At this point we were able to verify our correctness properties of this part of the Bluetooth protocol. Which we consider to be a very satisfactory result of the project. It is not something that has not been done before in this way and yet proved to be very useful.

3 Project report

To give a more illustrative explanation of what we have done, we have included three (slightly altered) sections of our report that resulted from the project. These will give an introduction to the topic, an informal description of part of Bluetooth we have modeled and the conclusions we have drawn using our model. A more detailed description can be found in this report, which can be found on our webpage [3].

3.1 Introduction to the topic

Bluetooth is a widely used communication protocol these days. It is used in the communication between phones, computers, headsets and many more devices. In the year 1994 the Ericsson company decided it wanted a protocol that could be used to connect mobile phones to other devices. Jaap Haartsen, working for Ericsson, developed the protocol. The techniques were further developed by the Bluetooth Special Interest Group.

One of the things described in the protocol is the way in which two devices that are neither connected nor synchronized can try to find each other. This is called the Inquiry Response Phase, the first phase in the protocol, which should provide a way for the devices to synchronize in order to allow further communication.

We have looked closer at the specification of this phase as described in [2] and created an UPPAAL model to formally verify that after the Inquiry Response Phase indeed the devices will be synchronized.

3.2 Informal description

When two Bluetooth devices want to start communicating they do that using the Inquiry Phases. In these phases one of the devices is assumed to be in master mode querying for other devices. The other device is assumed to be in slave mode. The master keeps sending packages and listening for responses. The slave will listen for a package from the master and respond to the master by sending a return package.

The devices do, however, change frequency during every phase. The frequencies used in Bluetooth are very common frequencies used in wireless phones, remote controls, garage doors and more. Therefore the devices change ("hop") their frequencies a lot. The devices are unlikely to use the same frequency the first time and the communication attempt will fail. However, the hopping should be done in a way that at a certain point in time the devices will use the same frequency in the same time interval and further synchronization can be achieved using that.

There are quite a few tricks involved in order to get this to work properly. There is the hopping of frequencies, timing issues in sending, receiving and listening and some more.

We want to verify that indeed the devices will eventually synchronize in all cases if we follow the specification. To do this we have constructed a UPPAAL model that represents the relevant bluetooth phases. As illustration we have added the part of this model that is responsible for the device execution. This can be found in figure 1.

3.3 Results and conclusions

We have created a model of which we think is sufficiently close to reality to be used in the verification of some properties of the Inquiry Response Phase.

Modeling the Inquiry Response Phase in UPPAAL worked rather well. It gave us good insight in the phase and some questions surfaced that we could not answer easily. We have even found a strange remark in the specification that to our opinion is incorrect.

Although it is arguable whether the specification is well written, at least we could, with some effort, all agree on what we think the specification specifies.

We verified that always eventually the master device will receive a return packet from the slave for a lot of initial values. This means we have a strong belief that two Bluetooth devices will eventually synchronize.

The UPPAAL model we have created can be downloaded from
`www.cs.ru.nl/ita/publications/papers/fvaan/bluetooth/`
The website of the UPPAAL project is
`http://www.uppaal.com`

3.3.1 Verification Results

In total, we have tried to prove two properties of the system, representing system liveness and safety. These are:

- $A\Diamond$ `Master.Finished` $\wedge$ `Slave.Finished`
This property actually expresses that the system always eventually will reach the "finished" state for both devices, i.e. it expresses that always eventually the master device will receive a return packet from the slave. Actually, this property is the desired property the developers of Bluetooth would want to satisfy under all conditions. We have validated this important property for a whole variety of initial clock values. Besides that, we have been able to verify these properties, too, for both ideal Bluetooth clocks as well as clocks that were subject to drift and jitter.

  Some of the validated configurations:

maxwaitbit	deviation (UPPAAL time units)	time (min)
4	1	1
7	1	3
7	3	8

  The `maxwaitbit` indicates that when bit number `maxwaitbit` of the Bluetooth clock is or becomes 1 the devices must enter the Bluetooth phase instead of waiting at the initial state. This way we can check a range of initial clock values. Ideally we should verify this for maxwaitbit being the maximum clock bit. But that simply takes too long.

  The `deviation` indicates that, for each period of 625 UPPAAL time units, the clock tick may differ this amount of UPPAAL time units. A deviation of 3 indicates therefore a deviation of $\frac{3}{625}$ UPPAAL time units, indicating a maximum deviation of about 7 minutes a day. In practice the clocks used will not be that bad, therefore this means in practice that synchronization is accomplished. We could have used a tighter interval but apart from resources required for the verification this would not affect the result.

- $A\Box$ `not deadlock`
  This property actually expresses a system invariant, stating that the system as a whole will never deadlock. This property is very important in getting a confidence that the specification is correctly modeled. Of course if the system can always reach the state `Finished` for all devices, it cannot have deadlocked.

  Some of the validated configurations:

maxwaitbit	deviation (UPPAAL time units)	time (min)
0	0	1
2	0	1
4	0	1
7	0	1

With these properties satisfied and no counter examples found we have a strong belief that in our model these properties actually hold for all initial values. Therefore we think in the Bluetooth protocol the devices will also find each other eventually.

4 Results

4.1 How our approach solved the problem

The idea of verifying a real world standard protocol instead of a textbook example is a motivation and an opportunity in the sense that it helps students recognise the values of the teaching in an applied system.

Graphical visualisation, modelling and simulation of a problem, help to understand formal methods through giving different dimensions of the formal methods other than the theoretical formulae.

The almost social atmosphere created by working in relatively small groups when dealing with a problem, enhances the learning process by creating an environment that allows the weak students to learn from the stronger ones. Sometimes a student may not feel free to participate actively in class but when in a small group he/she will feel more at ease to ask even the dumbest question.

A tool like UPPAAL that supports a versatile range of programming language formats, also is a solution in a way because as students try to reflect on their understanding of the formal methods in a piece of code in a less restricted environment, they achieve a deeper understanding of the methods and probably begin to like the exercise. Besides being a rich tool, UPPAAL also reduces the limitions that in many tools are enforced due to poor representation of automata states or boolean conditions.

4.1.1 Further Research and related work

Probably we could verify more situations than the ones we did verify thus far. So another group could work on that.

One of the things we didn't pay much attention to is the duration of a transmission in the Inquiry Response Phase. Currently we assume that a transmission, if the receiver listens in time, will arrive completely and without errors or not arrive at all. Time does not elapse while sending or receiving. In reality this is not the case and it might be something to take a closer look at.

Something else we did not look at, but probably will be relatively easy to do using the model is verify properties for more than two instances (master, slave, slave for example).

Our research focused on the Inquiry Response Phase, leaving out other phases. Obviously these could be interesting.

Closely related work can be found in the paper [4] where the probabilistic tool PRISM was used to analyze the Inquiry Phase. [4] focuses on the probabilistic behavior of the Bluetooth communication. Where this paper mainly looks at the expected and worst and best case timing issues, our research mainly focuses on whether the communication will actually be succesful or not.

5 Knowledge and Skills

5.1 Required knowledge and skills

Team work skills Most real world problems (protocols) are systems enormous in size. For students to profitably work on such systems there is a need for teams/groups thus calling for team work skills as an essential tool for this approach.

Mathematical background It requires a relatively good background of mathematics to a certain detail, to visualise and understand formal methods, and in relation to this, students may also need to be familiar with transition systems. And most importantly they should have a good knowledge of the tool being used for the simulation and modelling. All of this also requires basic programming/modelling skills/knowledge for the students to be able to reflect their understanding of the formal methods into modal code that leads the simulations.

5.2 Desired skills and knowledge outcomes

From the objectives of the course [1] we can select the objectives relevant for this assignment, which will be:

1. Being able to recognize situations in which the applications of formal methods for specification and verification may be useful.
2. Being able to model distributed algorithms and protocols (or more generally: reactive systems) as networks of automata.

3. Being able to formalize desired properties of these algorithms and protocols in terms of automata or temporal logic.

4. Being able to use state-of-the-art proof techniques and computer tools for the analysis of embedded systems and protocols of "average" complexity.

5.3 Outcome skills and knowledge

Let us finally check whether each objective in section 5.2 is met:

1. We are able to recognize some situations in which the applications of formal methods for specification and verification may be useful.

2. We are able to model a part of the Bluetooth protocol as networks of automata.

3. We are able to formalize desired properties of the Bluetooth protocol in terms of automata or temporal logic.

4. We are able to use UPPAAL for the analysis of a part of the Bluetooth protocol.

So we well meet the objectives for this assignment for the bluetooth protocol, which wasn't a very specific one. So we are probably well able to do the same project for other protocols.

References

[1] Analysis of embedded systems. URL http://www.cs.ru.nl/~fvaan/PV/.

[2] Baseband Specification. In *Bluetooth–Core Specification v2.0 + EDR*, pages 55–210. 2004. URL http://bluetooth.com/NR/rdonlyres/1F6469BA-6AE7-42B6-B5A1-65148B9DB238/840/Core_v210_EDR.zip.

[3] Supporting formal method teaching with real-life protocols website. URL www.cs.ru.nl/ita/publications/papers/fvaan/bluetooth/.

[4] M. Duflot, M. Kwiatkowska, G. Norman, and D. Parker. A Formal Analysis of Bluetooth Device Discovery. In *1st International Symposium on Leveraging Applications of Formal Methods (ISOLA'04).*, November 2004.

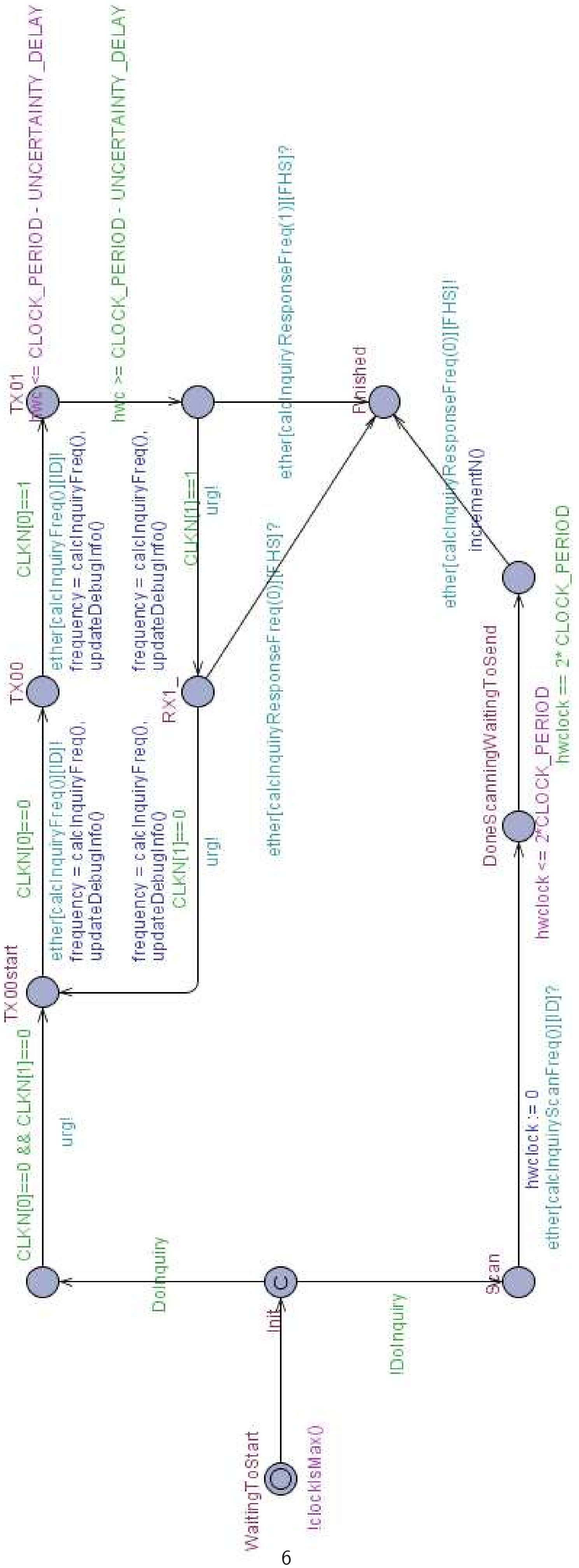
WaitingToStart
!clockIsMax()
Init
C
DoInquiry
!DoInquiry
Scan
CLKN[0]==0 && CLKN[1]==0
urg!
TX00start
CLKN[0]==0
ether[calcInquiryFreq()][ID]!
frequency = calcInquiryFreq(),
updateDebugInfo()
TX00
CLKN[0]==1
ether[calcInquiryFreq()][ID]!
frequency = calcInquiryFreq(),
updateDebugInfo()
TX01
hwc <= CLOCK_PERIOD - UNCERTAINTY_DELAY
hwc >= CLOCK_PERIOD - UNCERTAINTY_DELAY
frequency = calcInquiryFreq(),
updateDebugInfo()
CLKN[1]==1
urg!
RX1_
frequency = calcInquiryFreq(),
updateDebugInfo()
CLKN[1]==0
urg!
ether[calcInquiryResponseFreq(1)][FHS]?
ether[calcInquiryResponseFreq(0)][FHS]?
Finished
ether[calcInquiryResponseFreq(0)][FHS]!
incrementN()
hwclock := 0
ether[calcInquiryScanFreq()][ID]?
hwclock <= 2*CLOCK_PERIOD
DoneScanningWaitingToSend
hwclock == 2* CLOCK_PERIOD

Figure 1: The full Device template as modeled in UPPAAL