university of groningen

faculty of science and engineering

Enhancing the Automata Theory Course with Notions from Reactive Systems

Miriam-Bogdana Agafiței

university of groningen / faculty of science and engineering

University of Groningen

Enhancing the Automata Theory Course with Notions from Reactive Systems

Bachelor's Thesis

Bachelor of Science in Computing Science
at the University of Groningen under the supervision of
Prof. dr. Jorge Pérez
and
Dr. Revantha Ramanayake

Miriam-Bogdana Agafiței (s4162747)

August 29, 2022

Contents

Abstract

During the second year of the BSc Computing Science at the University of Groningen, students take a course called “Languages and Machines”. During this course students learn the fundamentals of Automata Theory, which is an active research area in Computer Science.

Unsurprisingly, there is a gap between the course’s content and recent developments in Automata Theory. There are extensions to classical Automata Theory that involve reactive systems which represent the reactive behavior of modern computing systems. This Bachelor project aims to help introduce “Languages and Machines” students to the recent reactive developments in Automata Theory by creating educational material.

1 Introduction

"Languages and Machines" is a course taught in the second year of the BSc Computing Science. This course provides an introduction to classical results in Automata Theory. Understandably, there is a gap between the course's content and recent developments in Automata Theory. We are interested in developments connecting Automata Theory and concurrency and interaction, which are two predominant phenomena in computing practice nowadays.

There are extensions to classical Automata Theory that go in the direction of mimicking the reactive behaviour of modern computing systems. Such extensions consider reactive systems which can be represented by abstract machines capable of performing non-terminating, interactive computations, rather than just terminating, non-interactive computations. These extensions bear a stronger resemblance to the computers we have nowadays. Of particular interest are Reactive Turing machines (RTMs), i.e., Turing machines (TMs) [1] that can interact with their environment by performing actions [2]. RTMs conservatively extend usual TMs; that is, usual TMs correspond to the class of RTMs that do not interact with their environment.

These newer reactive developments in Automata Theory come at a price: the features that allow interaction are more involved, and so they require more advanced knowledge to be understood. Indeed, most recent developments are still showcased solely in research papers that are written in technical language, which is inaccessible to BSc and MSc students. This means that if a second year "Languages and Machines" student would want to know more about the new and exciting concepts regarding reactive systems, they would struggle to grasp and appreciate these advanced models of computation.

1.1 Research Question

This Bachelor project is focused on bridging the gap between the "Languages and Machine" course's content and recent reactive developments in Automata Theory:

Therefore, the research question of this Bachelor Project is:

- How to introduce models of reactive systems to second-year Computing Science students?

1.2 Solution

Let us continue by presenting the solution that we found to the research question posed earlier. We believe that second-year Computing Science students can be introduced to models of reactive systems by extending the course lecture notes [3] (also known as the reader of the course) with a new chapter devoted to introducing notions about reactive systems, with a focus on RTMs.

RTMs stood out as a great choice for introducing the newer Automata Theory to second year Computing Science students, because they clearly showcase the recent direction that Automata Theory is taking, which involves a stronger focus on interaction. Furthermore, RTMs are an appropriate complement for students who have taken the "Languages and Machines" course because they bear many similarities to the TMs seen in the course.

The new chapter for the course's lecture notes was written for educational purposes, in a beginner-friendly way. It has the same style as other chapters in the reader, meaning that it includes: clear explanations and definitions, several examples with a gradual increase in difficulty, description of the main properties of reactive systems, a number of exercises to help consolidate the newly learned content, and solutions to the exercises.

Moreover, the new chapter about reactive systems is supplemented by a set of associated slides, to be used in the lectures. The slides contain a few animated examples of representation of reactive systems, such as RTMs. This will help present reactive systems in a dynamic and engaging way.

1.3 Thesis outline

In this paper, we will analyze the contributions created for the purpose of introducing models of reactive systems to second-year Computing Science students. After the introduction, we will move on to presenting a fair amount of background information. This includes some prerequisite knowledge, a general introduction of reactive systems, a state of the art analysis, and information about the key concepts that are involved in this project, such as RTMs, parallel composition and branching bisimulation.

We will continue by addressing the development process of this project which involved several different parts. We will address the learning process, as well as the process of developing the educational material. We will conclude with a summary of the contributions brought by this project and the possibilities of future work in this area.

2 Background

In this chapter we will start by presenting some prerequisite knowledge. Then, we will introduce reactive systems. Later, we will showcase a state of the art analysis regarding RTMs. We will continue by introducing the most important pieces of knowledge that are presented in the new reader chapter, such as RTMs, configurations, transitions systems, parallel composition, and branching bisimulation.

2.1 Prerequisites

Let us introduce some prerequisite knowledge which is needed in order to read the rest of this thesis. These notions are presented here with the assumption that the reader has some familiarity with finite state machines.

2.1.1 Turing Machines

We will start with a short introduction about TMs [1]. A TM is a finite state machine that is equipped with a tape that is divided into squares. A TM can write on the squares on the tape during a transition from a state to another state. The head of the machine can move to the right or to the left on the tape, which allows it to read and manipulate the input as desired. A TM is allowed to access and modify any memory position and it has no limitation on the space or the time available for a computation.

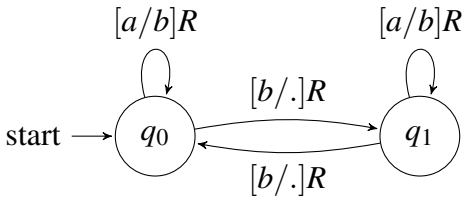


Figure 1: A TM that turns all a characters into b characters.

The TM shown in Figure 1 will have an input string on its tape formed out of a and b characters prior to starting its computation. The self-loop in state q_0 allows it to read a characters, replace them with b characters whilst the head of the machine moves to the right. If the machine encounters a b character on the tape, it will leave it unchanged whilst the head of the machine moves to the right and the machine transitions to state q_1.

In state q_1 there is a self-loop during which the machine can read a characters and replace them with b characters whilst the head of the machine moves to the right. If the machine encounters a b character on the tape, it will leave it unchanged whilst the head of the machine moves to the right and the machine transitions back to the starting state q_0. The machine will terminate when it reaches the end of the input string formed out of a and b characters on the tape.

2.1.2 Labelled transition systems

Labelled transition systems represent another piece of prerequisite knowledge. A labelled transition system is a useful type of notation that abstracts the structure of state based systems. It usually involves just the states and the transitions of the system, as well as the labels of the transitions. This notation is frequently used in Theoretical Computing Science.

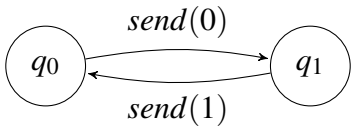


Figure 2: A labelled transition systems that sends 0 and 1 characters.

In Figure 2, the labelled transition system sends a 0 character during a transition from state q_0 to state q_1. When transitioning from state q_1 to state q_0 this labelled transition system sends a 1 character.

2.2 Reactive systems

Since reactive systems are a pivotal part of this project, we will address what reactive behavior means by taking a look at an example of a real-life reactive system. Then, we will continue by analyzing the importance of reactivity.

2.2.1 Example of a real-life reactive system

A simple example that can illustrate reactive behavior is a vending machine. A vending machine can interact with its users by reacting to some stimuli. For example, a vending machine can be informed of a user's particular snack choice when the user presses the button that signifies the desired snack. The vending machine will react to the user's choice by giving the chosen snack to the user, after the user pays.

Figure 3
Example of a real-life reactive system: a vending machine

Moreover, a vending machine is a non-terminating system because it does not close after it finishes an interaction with a user, it stays open so that it can be ready when another user to comes along.

2.2.2 The importance of reactivity

Now that we have touched upon the meaning of reactive behaviour, it is important to peruse the question:

- Why should BSc Computing Science students learn about models of reactive systems?

First of all, reactivity is a relevant topic nowadays because the majority of modern technology exhibits reactive behaviour. For example, technology that a lot of people have had some experience with, such as computers and smartphones, are reactive systems.

Students are introduced to TMs [1] during the "Languages and Machines" course. Although TMs

have been regarded as the fundamental paradigm for computing ever since Alan Turing developed the concept of machine computation, computing in the modern day encompasses much more than it used to. For example, not all of the input might be available prior to the beginning of computation, and some systems can operate indefinitely, whilst still acting correctly. TMs are not capable of such behaviour and as a result, they are not suitable for representing interaction. Since the interaction available in technology nowadays goes beyond the models presented in the "Languages and Machines" course, it is important to introduce students to the expressiveness of more recent reactive models.

Moreover, reactivity is important because it is a concept that is familiar for many people, including students. This is due to not only the fact that a lot of people are getting used to interacting with technology, but also due to people being used to interacting with each other. It is common for human interaction to involve one person reacting to another person [4].

Lastly, notions about reactive systems can be gradually introduced from the concepts that are already taught in the "Languages and Machines" course. In our opinion, learning about reactivity by focusing on RTMs will not represent a huge leap in knowledge for the students because RTMs bear many similarities to the TMs which are already featured in the course.

2.3 State of the Art

Let us continue by briefly exploring the state of the art regarding RTMs. RTMs were proposed as an extension to TMs in a research paper in 2011 by Jos Baeten, Bas Luttik, and Paul van Tilburg [5] and more thoroughly analyzed by them in a later paper in 2013 [2].

RTMs have been touched upon in a research paper that involves subjects within Automata Theory such as Context-Free Grammars and Pushdown Automata [6]. RTMs were also mentioned in a paper that aims to provide a uniform study about different notions regarding effectful state machines [7]. Moreover, RTMs have been compared to another type of TM called an Interactive Turing machine in paper [8].

RTMs have been analyzed in a paper that discusses the π-calculus [9]. The π-calculus is a formalism for expressing reactive and concurrent computation. The reason why π-calculus and RTMs were perused together is that π-calculus can act as a sort of programming language counterpart to RTMs [2]. RTMs are the sole focus of only two research papers: [5], and [2], where the explanations about them are not written in a way that would be easily accessible to BSc students.

All in all, the state of the art analysis shows that even though a number of papers have touched upon RTMs, none of the information available about them is presented in a way that could be easily grasped by BSc students. As a result, it seemed to be of value to create educational material that involves RTMs.

2.4 Reactive Turing machines

We will continue by introducing some of the key notions that are presented in the new reader chapter and the set of associated slides. We will start by presenting RTMs [2]. As stated beforehand, RTMs are an extension to TMs which is capable of interaction and non-termination.

An RTM is a reactive system that is able to operate just as a TM would whilst also having the ability to perform an action during each transition. An RTM can perform two different types of actions:

- observable actions that involve reacting to its environment or other RTMs
- actions that are unobservable to its environment or other RTMs. Such actions entail solely the internal computations that a TM is also capable of doing during each transition.

A TM can perform only unobservable actions. As a result, a TM is an RTM where all the transitions are unobservable. Unobservable actions are denoted by the Greek letter τ.

In Figure 4 we will show how the TM from Figure 1 would look like with the notation used for an RTM:

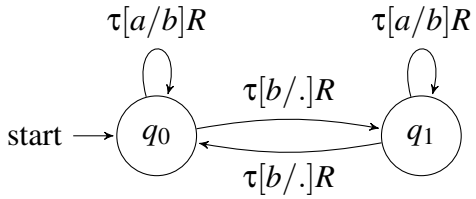


Figure 4: The TM from Figure 1 shown with the notation used for an RTM. As before, this machine turns all *a* characters into *b* characters.

An RTM can perform both observable and unobservable actions. Let us dive deeper into how observable actions work. Observable actions involve communication between RTMs or RTMs and their environment. This communication is done via designated *data transferring channels*. The following notation is used for observable actions:

- $c!d$ signifies that an RTM sends some *data d* via *channel c* to its environment or another RTM
- $c?d$ signifies that an RTM receives some *data d* via *channel c* from its environment or another RTM

Let us take a look at a small example of an RTM that performs the same internal computations as the TM from Figure 1, as well as two observable actions in which it sends *a* characters via the communication channel *i*.

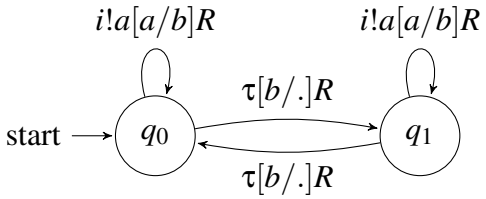


Figure 5: An example of an RTM that turns all *a* characters into *b* characters and sends all *a* characters from the tape via channel *i*. We will name this RTM RTM_1.

The RTM shown in Figure 5 will have an input string on its tape formed out of *a* and *b* characters prior to starting its computation. The self-loop in state q_0 allows it to read *a* characters, replace them

with b characters whilst the head of the machine moves to the right, as well as perform the observable action of sending a characters via channel i. If the machine encounters a b character on the tape, it will leave it unchanged whilst the head of the machine moves to the right and the machine makes an unobservable transition to state q_1.

In state q_1 there is a self-loop during which the machine can read a characters and replace them with b characters whilst the head of the machine moves to the right, as well as perform the observable action of sending a via channel i. If the machine encounters a b character on the tape, it will leave it unchanged whilst the head of the machine moves to the right and the machine makes an unobservable transition back to the starting state q_0.

Although we have seen that the RTM from Figure 5 has some added capabilities to the TM in Figure 1, it might not fully clear at this point how exactly RTMs showcase their reactive abilities. In order to clear this up, we will take a look at an example of another RTM that has access to the communication channel i. The RTM shown in Figure 6 will receive the a characters sent by the RTM in Figure 5 via channel i and place each of them on the tape followed by a $*$ character.

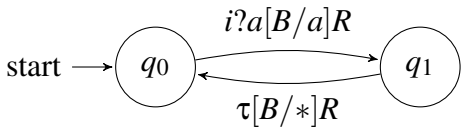


Figure 6: An example of an RTM that receives the a characters sent via channel i by RTM_1 from Figure 5. This RTM places each a character on the tape followed by a $*$ character. We will name this RTM RTM_2.

RTM_2, which is shown in Figure 6 will receive as input an a character during the observable transition from state q_0 to state q_1. During this transition it will place the a character it has received on a blank spot on the tape (denoted by B), whilst the head of the machine moves to the right.

The transition from q_1 to q_0 is unobservable. During this transition RTM_2 will place a $*$ character to the right of the a character that was just placed on the tape in the previous transition, whilst the head of the machine moves to the right. This transition causes the machine to reach the starting state, which means that RTM_2 is ready to receive input again. This RTM is non-terminating, it will always be ready to receive input via channel i.

2.4.1 Configuration

The configuration of an RTM [2] is another important concept that is showcased in the new reader chapter and the slides. This concept is relevant to RTMs so we shall continue by succinctly explaining what a configuration represents.

The global state of an RTM can be determined by a configuration. A configuration will encode information regarding:

1. a state of the machine
2. a tape instance

A tape instance is responsible for encoding the contents of the tape and the position of the read/write head. It is represented by the Greek letter δ.

An example of how a transition between two states of an RTM looks like encoded in a configuration:

$$q_1 \xrightarrow{i!a[0/1]R} q_2 \Leftrightarrow (q_1, \delta_1) \xrightarrow{i!a} (q_2, \delta_2), \text{ where}$$

$$\delta_1 = \delta_L \check{0}\ 2\ \delta_R \text{ and}$$

$$\delta_2 = \delta_L\ 1\ \check{2}\ \delta_R$$

$\delta_1 = \delta_L \check{0}\ 2\ \delta_R$ signifies that the head of the machine has the symbol 0 underneath and that the symbol to the right of 0 on the tape is 2.

$\delta_2 = \delta_L\ 1\ \check{2}\ \delta_R$ signifies that the symbol 0, which was under the head of the machine in the configuration δ_1 has been replaced by the symbol 1 on the tape and that the head of the machine has moved to the right. Now the symbol found to the right of 1 (which happens to be the symbol 2) is under the head of the machine.

2.4.2 Transition system

The notion of a transition system associated with an RTM [2] is another significant notion discussed in the new reader chapter. Let us explain in short how a transition system associated with an RTM functions.

For an RTM M, the associated transition system is denoted by $T(M)$. A transition system associated with an RTM is meant to highlight only the basic structure of the RTM. In order to achieve this, configurations are used. A transition system abstracts some of the details of the transitions between states by using configurations. The states of a $T(M)$ are the configurations of the RTM M.

In Figure 7 we will show the *transition system* $T(2)$ which is the associated transition system of the RTM_2 from Figure 6.

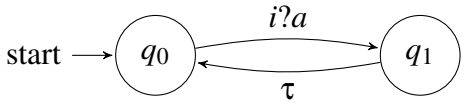


Figure 7: The *transition system* $T(2)$ which is the associated transition system of the RTM_2 from Figure 6

As we can see, some of the information that was found on top of the transitions of RTM_2 is now encoded in configurations.

$$q_0 \xrightarrow{i?a[B/b]R} q_1 \text{ becomes } q_0 \xrightarrow{i?a} q_1$$
$$q_1 \xrightarrow{\tau[B/*]R} q_0 \text{ becomes } q_1 \xrightarrow{\tau} q_0$$

2.5 Parallel composition

Another notable concept that is presented in the new reader chapter is parallel composition [2]. We will continue by giving a brief overview of how parallel composition works. Parallel composition helps us illustrate how RTMs run in parallel and interact with each other.

With RTM_1 (Figure 5) and RTM_2 (Figure 6) we have seen how a simple interaction could occur between two RTMs. We can formalize the interaction of two RTMs with the notion of parallel composition. Given that RTM_1 and RTM_2 are two RTMs that both have access to the communication channel i then, the parallel composition of RTM_1 and RTM_2 on the communication channel i exhibits the behaviour of outputting the string $[a*]^m$ on the tape of RTM_2, where $m \in \mathbb{N}$, and m signifies the number of a characters that RTM_2 receives as input via channel i from RTM_1.

Therefore, we can see that parallel composition is useful for defining the behaviour of two RTMs that are interacting with each other.

2.6 Branching bisimulation

Branching bisimulation [2] is the last concept featured in the new reader chapter that we will address in this thesis. The concept of a branching bisimulation is used in order to formally compare the behaviour of transition systems. It helps in deciding if two transition systems are behaviourally equivalent. The notion of branching bisimulation is relevant in the context of RTMs because we can have two different RTMs whose underlying transition systems are branching bisimilar. This means that we can compare the behaviour of RTMs by analyzing their transitions systems to see if branching bisimilarity exists.

If two transitions systems are behaviourally equivalent they will be called branching bisimilar. A branching bisimilarity exists between two transition systems if they both perform the same observable actions overall and reach states where the same choices are possible. Only observable behaviour is taken into consideration. We will abstract from the internal behavior. Each transition system can have different internal computations compared to the other one.

We will take a look at two RTMs that have transition systems that are branching bisimilar.

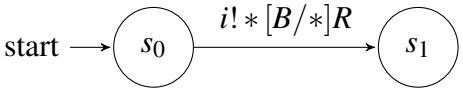


Figure 8: An RTM that performs one observable action in which it sends a $*$ character via the communication channel i. We will call this RTM RTM_3

We can see that RTM_3 has one observable transition between state s_0 and state s_1 in which it sends a $*$ character via a communication channel called i while also placing a $*$ character onto the tape and moving the head to the right.

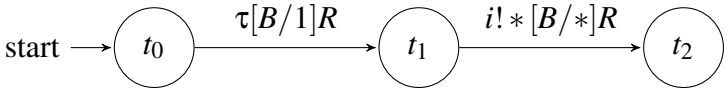


Figure 9: An RTM that performs one unobservable transition and one observable action in which it sends a $*$ character via the communication channel i. We will call this RTM RTM_4

RTM_4 has an unobservable transition between state t_0 and state t_1 during which it places a 1 character onto the tape and moves the head to the right. Then, RTM_4 has one observable transition between state t_1 and state t_2 in which it sends a $*$ character via a communication channel called i while also placing a $*$ character onto the tape and moving the head to the right.

We can see that RTM_3 (Figure 8) and RTM_4 (Figure 9) perform the same observable action but not the same unobservable actions. The fact that RTM_4 performs an unobservable transition that RTM_3 does not perform, does not prevent their underlying transitions systems from being bisimilar. What matters is that both RTMs perform the same observable action of sending a $*$ character via the communication channel i.

Let us take a look at the two transition systems of the RTMs shown in Figure 8 and Figure 9. These two transition systems are branching bisimilar. They will be shown in Figure 10 and 11.

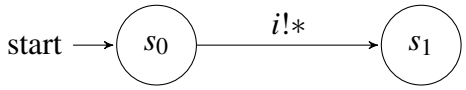


Figure 10: The transition system associated with RTM_3 (Figure 8). This transition system performs one observable action in which it sends a $*$ character via the communication channel i. We will call this transition system $T(3)$.

The transition system in Figure 10, $T(3)$ has one observable transition between state s_0 and state s_1 in which the observable action of sending a $*$ character via the communication channel i is performed.

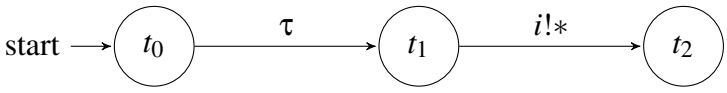


Figure 11: The transition system associated with RTM_4 (Figure 9). This transition system performs an unobservable action, as well as one observable action in which it sends a $*$ character via the communication channel i. We will call this transition system $T(4)$.

The transition system in Figure 11, $T(4)$ has a transition between state t_0 and state t_1 where an unobservable action τ is performed. Then, it has one observable transition between state t_1 and state t_2 in which action a is performed.

$T(3)$ and $T(4)$ are branching bisimilar because the observable transition $i!*$ that $T(3)$ performs is also performed by $T(4)$. It is alright that $T(4)$ performs some internal computation in an unobservable transition before it performs action $i!*$.

3 Development process

In this chapter we will analyze the development process of this project. We will start with the learning part of the project and reiterate what were the most important parts of the acquired knowledge. Then, we will discuss the process of coming up with the chapter and the set of associated slides by focusing on the key ideas that were followed when producing the new educational material.

3.1 Learning

Before beginning to write the new reader chapter dedicated to reactive systems and creating the slides associated to the chapter, it was necessary to thoroughly understand a lot of concepts about reactive systems.

As stated beforehand, the recent developments in Automata Theory that involve reactivity that we aim to present to BSc students in the new reader chapter are showcased solely in research papers that are written in technical language, which is inaccessible to BSc students. This meant that the task at hand quite was quite difficult and lengthy.

The starting point in the learning journey was becoming familiar with the idea of connecting Automata Theory with concurrency and interaction, which are two predominant phenomena in computing practice nowadays. This was achieved by studying "Elements of interaction: Turing award lecture" by Robin Milner [10].

The biggest milestone in the learning process was understanding the concepts presented about RTMs in the research paper "Reactive Turing machines"[2]. Understating how RTMs are defined and several other important concepts surrounding them, such as the transition system associated with an RTM, parallel composition and branching bisimulation was of the utmost importance. Once this was achieved, we could move on to approaching the creation of the new educational material.

3.2 Creating the material

In this section we will present the process of coming up with the new educational material:

- the chapter
- the slides associated with the chapter

We will also describe what were the key ideas that were followed when producing the material.

3.2.1 The chapter

After understanding the concepts about reactive automata which would be included in the new reader chapter, it was time to approach writing it. In order to ensure that the new chapter did not feel out of place in the reader next to the pre-existing chapters, we aimed to take a similar writing style in the new chapter about reactive systems.

A key idea that we followed when producing new chapter was building on top of the knowledge that the students have at the end of the "Languages and Machines" course. We pursued this because

we believed it was crucial to slowly present the new concepts on top of the foundation of knowledge that the students have so that they can have an enjoyable learning experience. In order to achieve this, we did the following:

- We referred back to examples or definitions from a previous chapter in the reader [3] whenever possible to ensure that students can easily link the old information to the new one.
- We carefully analyzed what knowledge the students have at the end of the "Languages and Machines" course and tried to find the smoothest transition for introducing RTMs.

Let us go more in depth about how we achieved the latter. At the end of the "Languages and Machines" course students know that there are many classes of machines for recognizing strings and that TMs are the most expressive one.

Since we wanted to introduce a new type of TM to the students, we had explain why that would be necessary. As a result, the new chapter begins with a presentation of the shortcomings of TMs.

The chapter continues by presenting the idea of reactive systems. We thought about directly introducing RTMs at this point but we decided that it would be too abrupt of a leap in knowledge. We reached this decision especially because RTMs are defined with the help of a labelled transition system in paper [2], which is concept that is not covered in previous chapters of the reader [3].

As a consequence, the chapter continues by introducing labelled transition systems and showcasing how reactive systems can be represented by labelled transition systems. Next in the chapter is the way in which two reactive systems represented by labelled transition systems can interact with each other.

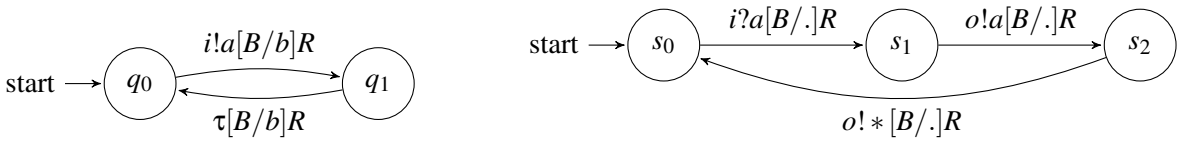
After this, the first section of the chapter ends. We decided to end each section of the chapter with a subsection called "Takeaways". In this section the key concepts from the section are reiterated in a bullet point list. This was done to help students remember the most important parts from each section as they go through the chapter and move on to another section. We took the inspiration of having a "Takeaways" section from the book [11].

From this point onward, the chapter structure resembles the order in which concepts are introduced in the paper [2]. The second section of the chapter focuses on introducing RTMs, configurations and transition systems associated with RTMs. The central point of the third section is parallel composition. The fourth section is dedicated to presenting branching bisimulation.

Another key idea that we followed when producing the new chapter was having plenty of examples for each of the newly introduced topics. We strived to achieve this because it has been scientifically proven that having examples for newly learned topics is a beneficial learning strategy [12]. During each of the sections described above, there are several examples. The examples slowly increase in difficulty so that students do not have a hard time understanding them.

Most examples are found in the parallel composition section. This is due to the fact that we felt that the way RTMs interact by running in parallel was one of the most important notions that students should understand after reading the chapter.

Let us take a look at some examples of RTMs that were created in order to highlight parallel composition in the new reader chapter. We will be able to see how the examples slowly increase in size and difficulty. We will start with the first example of parallel composition that is shown in the chapter.

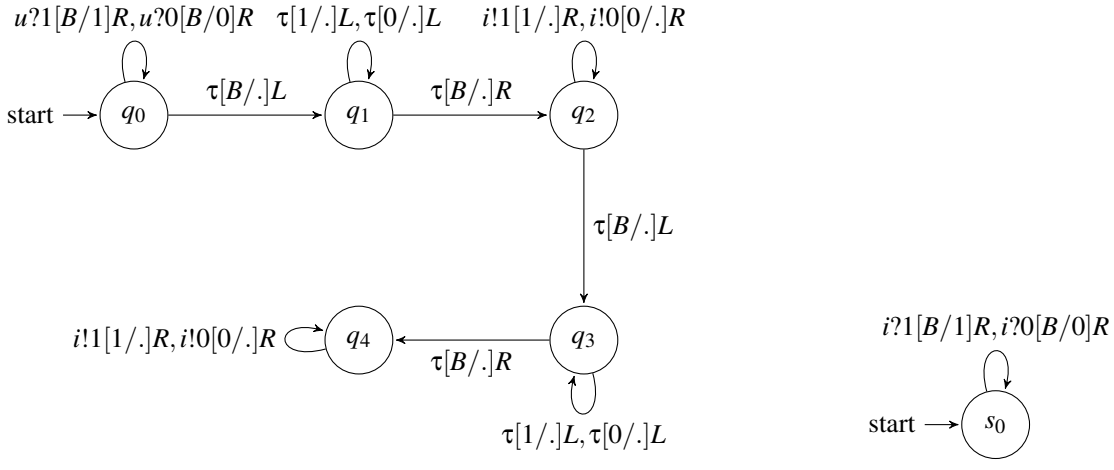


(a) RTM_5 (b) RTM_6

Figure 12: The first example of parallel composition that was created for the chapter is quite small and has a low level of difficulty. This was done in order to gently introduce students to this new concept.

The example in Figure 12 shows two RTMs interacting with each other. This example was designed to be quite small and easy to follow. We wanted to ensure that the students are slowly introduced to the concept of parallel composition.

In Figure 12, the RTM on the left (RTM_5) continuously sends a characters via channel i. The RTM on the right (RTM_6) receives the a characters and outputs each of them followed by a $*$ character to the environment via channel o. Therefore, given that RTM_5 and RTM_6 are two RTMs that both have access to the communication channel i then, the parallel composition of RTM_5 and RTM_6 on the communication channel i exhibits the behaviour of outputting the string $[a*]^n$ on channel o, where $n \in \mathbb{N}$, and n signifies the number of a characters that RTM_6 receives as input via channel i from RTM_5. Let us continue by taking a look at an example of parallel composition that is shown at a later point in the chapter.

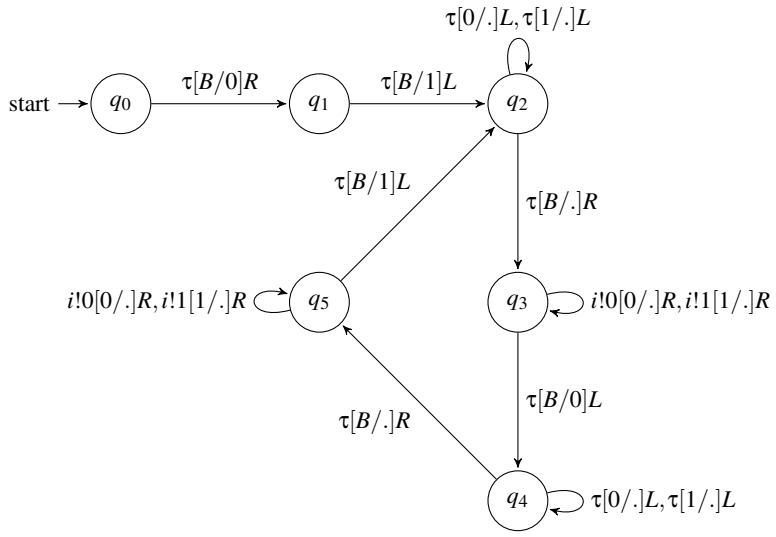


(a) RTM_7 (b) RTM_8

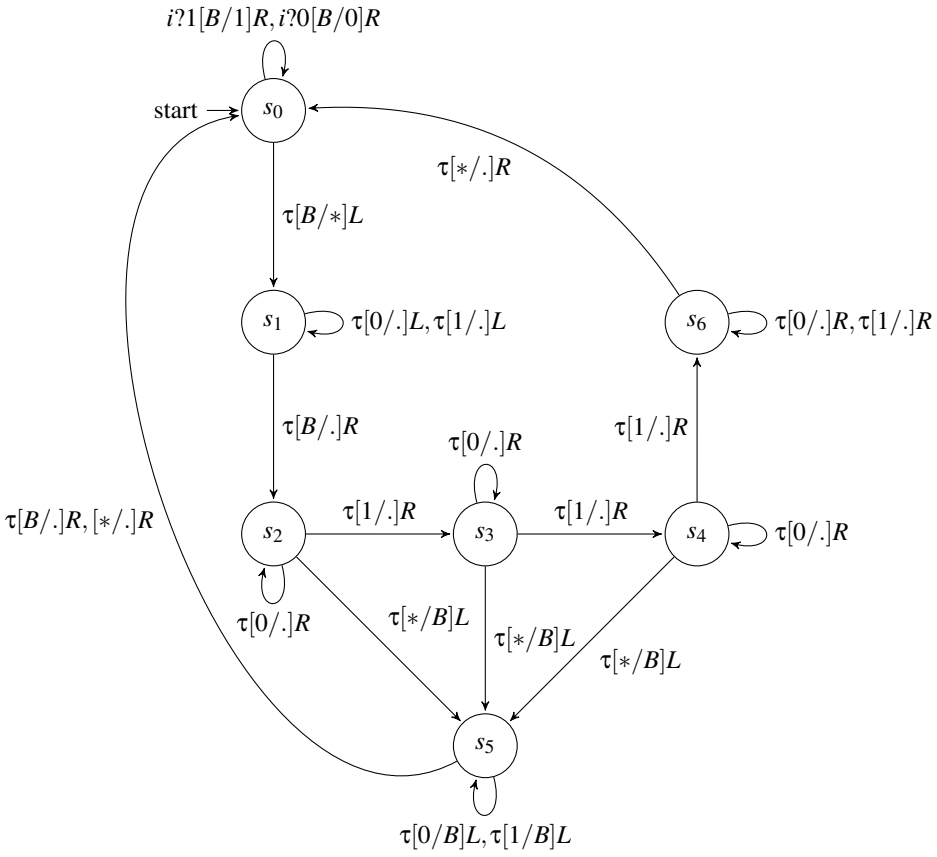
Figure 13: An example of parallel composition that is presented at later point in the chapter. This example was designed to be slightly more involved and have a medium level of difficulty.

In Figure 13, the RTM on the left (RTM_7) will receive input via channel u from its environment and it will place it onto its tape. RTM_7 will then go through the contents of its tape twice and both times it will send the contents to the RTM on the right (RTM_8) via channel i. RTM_8 will receive the input string twice from RTM_7 and place it onto its tape. Therefore, the parallel composition of RTM_7 and

RTM_8 on the communication channel i exhibits the behaviour of outputting the doubled version of the input string on the tape of RTM_8. We will move on by taking a look at the last parallel composition example that is shown in the chapter.



(a) RTM_9



(b) RTM_{10}

Figure 14: The last example of parallel composition that is presented in the chapter is the biggest and most involved on compared to the previous ones.

In Figure 14 RTM_9 generates strings made out of 0 and 1 characters. The first string that it sends via channel i is 01, the next one will be 010, followed by 0101 and so on. RTM_{10} receives these strings and keeps on its tape all the strings that have more than three 0 characters. Therefore, given that RTM_9 and RTM_{10} are RTMs that both have access to the communication channel i, the parallel composition RTM_9 and RTM_{10} on the communication channel i exhibits the behaviour of outputting the string $01010 * 010101 * 0101010 * 01010101...$ on the tape of RTM_{10}.

The last section of the chapter is dedicated to exercises that were created with the purpose of consolidating the newly learned content. Each exercise tests a particular part of the key concepts that were described in the chapter, such as labelled transition systems, RTMs, the transition system associated with an RTM, parallel composition and branching bisimulation. The exercises were built with a similar level of difficulty as the examples seen in the chapter. Moreover, exercises on the same topic gradually increase in difficulty. It is also worth noting that solutions were developed for all of the exercises.

3.2.2 The slides associated with the chapter

After finishing writing the chapter, it was time to approach creating the slides associated with it. In this section, we will describe this process.

The first part of this process involved creating an outline of all the key topics that would be covered in the slides. This was followed by summarizing the most important concepts surrounding each of these topics in brief bullet point lists that could be used for the slides.

Then, it was time to approach the examples for the key topics covered in the slides, such as labelled transition systems, RTMs, the transition system associated with an RTM, parallel composition and branching bisimulation. It was necessary to adapt or even create new examples from scratch that differ from the ones made for the chapter in order to ensure that the examples on the slides would be appropriate in difficulty to be shown in a lecture. We did not want the examples to be too involved because we our aim was for students to be able to follow the lecture without much difficulty. Consequently, smaller examples were used when making the slides.

Another important feature of the examples in the slides is the fact that they are animated. The feature of overlays from the Beamer class was used in order to create the aspect of animation when going through the slides.

The examples in the slides were animated so that students could visualize the most difficult concepts, in particular parallel composition which might be trickier to picture without an animation compared to other notions since it involves picturing four different elements (two RTMs and two tapes, one for each RTM). In the animated examples of parallel composition, two RTMs were shown side by side with their respective tapes underneath. Both the RTMs and their tapes would appear to be animated when going through the slides.

4 Conclusion

4.1 Summary of Main Contributions

The goal of this bachelor thesis is to enhance the "Languages and Machines" course with notions about reactive systems. RTMs represent a specific way of introducing concepts related to reactive systems. The aim was to introduce students to the new and exciting parts of Automata Theory that go in the direction of mimicking the reactive behaviour of modern computing systems. In order to meet this goal:

- A new chapter for the "Languages and Machines" reader was written about several important notions regarding reactive systems such as labelled transition systems, RTMs, the transition system associated with an RTM, parallel composition and branching bisimulation. This chapter was designed to follow the style of the other chapters from the reader.
- A set of slides associated with the new chapter was created for the purpose of being used in the lectures. This set of slides includes animations that will help present reactive systems in a dynamic and engaging way.

4.2 Future Work

In this section we will discuss what future work may be done in the area of this project.

It could be of value to perform some user testing on a small group of students (perhaps 5 to 10 students) who have completed the "Languages and Machines" before introducing the new educational material in the next academic year. After the sample group of students gets introduced to the new material, their feedback could be incorporated by making some adjustments to the new reader chapter and the set of associated slides before they are used in the next academic year.

Moreover, in order to ensure that the new material performs well it should be carefully assessed according to the feedback given by the first generation of students that will be introduced to it. The feedback given by the first generation of students exposed to the new content should be mindfully incorporated for the following academic year.

Lastly, a re-evaluation of the state of the art in Automata Theory could be done once every few years. This might help with ensuring that students always get introduced to valuable and relevant recent Automata Theory content alongside the fundamentals of Automata Theory during the "Languages and Machines" course.

Bibliography

[1] A. Turing, "On computable numbers, with an application to the n tscheidungsproblem," *Proceedings of the London Mathematical Society*, vol. 42, no. 1, 1936.

[2] J. C. Baeten, B. Luttik, and P. van Tilburg, "Reactive turing machines," *Information and Computation*, vol. 231, pp. 143–166, 2013. Fundamentals of Computation Theory.

[3] W. Hesselink, "Lecture notes: Languages and machines," 2014.

[4] D.-A. Huang and K. M. Kitani, "Action-reaction: Forecasting the dynamics of human interaction," in *European Conference on Computer Vision*, pp. 489–504, Springer, 2014.

[5] J. Baeten, B. Luttik, and P. v. Tilburg, "Computations and interaction," in *International Conference on Distributed Computing and Internet Technology*, pp. 35–54, Springer, 2011.

[6] J. Baeten, C. Carissimo, and B. Luttik, "Pushdown automata and context-free grammars in bisimulation semantics," in *9th Conference on Algebra and Coalgebra in Computer Science (CALCO 2021)*, Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[7] S. Goncharov, S. Milius, and A. Silva, "Toward a uniform theory of effectful state machines," *ACM Transactions on Computational Logic (TOCL)*, vol. 21, no. 3, pp. 1–63, 2020.

[8] B. Luttik and F. Yang, "On the executability of interactive computation," in *Conference on Computability in Europe*, pp. 312–322, Springer, 2016.

[9] B. Luttik and F. Yang, "The π-Calculus is Behaviourally Complete and Orbit-Finitely Executable," *Logical Methods in Computer Science*, vol. Volume 17, Issue 1, Feb. 2021.

[10] R. Milner, "Elements of interaction: Turing award lecture," *Communications of the ACM*, vol. 36, no. 1, pp. 78–89, 1993.

[11] S. Weinschenk, *100 things every designer needs to know about people*. Pearson Education, 2011.

[12] D. Tempelaar, B. Rienties, and Q. Nguyen, "Investigating learning strategies in a dispositional learning analytics context: the case of worked examples," in *Proceedings of the 8th International Conference on Learning Analytics and Knowledge*, pp. 201–205, 2018.

Acknowledgments

I would like to thank my supervisors Jorge Pérez and Revantha Ramanayake for providing guidance and explanations for some of the most difficult concepts involved in this project. Furthermore, I would also like to thank Jorge Pérez for introducing me to the subject of Automata Theory during my second Bachelor year during the “Languages and Machines” course and for introducing me to the interesting and exciting reactive part of Automata Theory for this project.