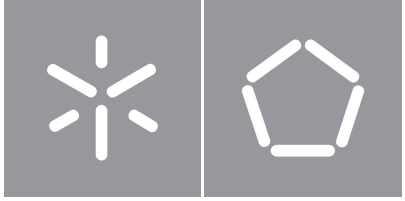
University of Minho
School of Engineering

Carlos Filipe Coelho Ferreira

Tackling Real-World Problems with Lince

february 2023



University of Minho
School of Engineering

Carlos Filipe Coelho Ferreira

Tackling Real-World Problems with Lince

Masters Dissertation
Master's in Informatics Engineering

Dissertation supervised by
Renato Jorge Araújo Neves
José Miguel Paiva Proença

february 2023

Abstract

Hybrid programs orchestrate classic computation with physical processes in order to reach prescribed goals. Examples of this are diverse and range from small medical devices, such as pacemakers and insulin pumps, to autonomous vehicles and district-wide electric/water grids. Their rapid development in the last decades lead to a flurry of research on suitable languages, semantics, and tools for their design and analysis.

Even though great progress has been made on this area, several important challenges remain largely open. A prominent case is that many crucial hybrid systems have not been tackled nor fully analysed via hybrid programming techniques and tools such as Lince. In fact, the area of hybrid programming is quite new and still generally lacks well-tested methodologies and techniques for a disciplined modelling/analysis of hybrid programs.

The goal of this project is to introduce a rich taxonomy of hybrid programs, and subsequently use it to fully explore the recently introduced tool Lince, its strengths and limitations. This will be a basis for further improving the tool. In our exploration, we will focus on those hybrid programs that are of crucial importance to society, typically found e.g. in medicine, autonomous piloting, and the energy sector. Such an exploration will not only clarify the potentialities and limitations of Lince, it will also provide interesting solutions to real-world problems in 'hybrid systems engineering'. Also, the fact that our exploration will be systematic and based on a previously established taxonomy means that it has the potential to help delineate previously unknown methodologies and techniques in hybrid programming.

Keywords Cyber-Physical Systems, Hybrid Systems, Hybrid Programming, Lince

Resumo

Programas híbridos orquestram métodos de computação clássica com processos físicos para alcançar certos objectivos previamente planeados. Exemplos de tais sistemas são diversos e vão desde pequenos dispositivos médicos, como *pacemakers* e bombas de insulina, até veículos autónomos e redes elétricas-/água. O seu rápido desenvolvimento nas últimas décadas levou a uma intensa investigação na área das linguagens, semântica e ferramentas adequadas para o seu desenvolvimento e análise.

Ainda que grandes progressos tenham sido feitos nesta área, vários desafios importantes permanecem em aberto. Um caso proeminente é que muitos sistemas híbridos cruciais não foram abordados nem totalmente analisados através de técnicas de programação híbrida e ferramentas como o Lince. Na verdade, a área da programação híbrida é bastante nova e ainda carece de metodologias e técnicas bem testadas para uma modelação e análise sistemática de programas híbridos.

O objetivo deste projeto é introduzir uma taxonomia rica de programas híbridos e, posteriormente, usa-la para explorar a ferramenta recém-lançada Lince, os seus pontos fortes e limitações. Esta será ainda uma base para melhorar a ferramenta. Nesta exploração, iremo-nos concentrar nos programas híbridos de importância crucial para sociedade, normalmente encontrados, por exemplo, em medicina, condução autónoma e no setor de energia. Tal exploração não apenas esclarecerá as potencialidades e limitações do Lince mas também fornecerá soluções interessantes para problemas do mundo real no campo da 'engenharia de sistemas híbridos'. Além disso, o facto da exploração ser sistemática significa que tem o potencial de ajudar a delinear metodologias anteriormente desconhecidas e técnicas dentro da programação híbrida.

Palavras-chave Sistemas Ciberfísicos, Sistemas Híbridos, Programação Híbrida, Lince

Contents

List of Figures

List of Tables

Abreviations

CPS Cyber-physical System.

PTR Pre-Thesis Report.

SA State of Art.

Glossary

Continuous System Dynamic system whose state flows according to a continuous function over time.

Cruise Control System that maintains a vehicle's previously programmed driving speed.

Cyber-physical System System with cybernetic and physical components to solve problems that neither party could solve alone.

Differential Equations Equations that contain the derivatives of one or more dependent (velocity) variables with respect to one or more independent (time) variables.

Discrete System Dynamic system whose state changes at given time instants.

Hybrid Automaton extension of classical automata with state variables that evolve continuously while in a state and change discretely at state transitions.

Hybrid Program Hybrid system specifications written in a programming-oriented style.

Hybrid System Mathematical models of dynamical systems that combine discrete and continuous dynamics.

Lince Hybrid program interpreter originating from the extension of differential equations to a classical while-language.

Part I

Introductory material

Chapter 1

Introduction

Key Points:

- Cyber-physical systems have great potential in all sectors of society.
- The development of these systems is recent as are the tools involved in such.
- The tool Lince has the ability to help in the development of these systems.
- The main objective of this dissertation is to contribute to the critical analysis of the tool through the development of a diverse taxonomy of hybrid programs.

1.1 Motivation and context

Programming and technology in general is becoming more and more intrinsic to the physical world as new ideas for possible applications of it emerge frequently in order to help solve problems once considered impossible. This trend, however, comes with its own set of problems that are difficult to handle and require much improvement and continued attention – computer science is still a long way from being able to represent the complexity of the physical world in a confident and reliable way. The reality around us is governed by a complex and connected set of numerous factors like time, space, randomness and many others that our cybernetic systems must take in special consideration to achieve the goals assigned to them, making them able to make decisions in a confident and safely way, similar to or better than what a human being would be able to.

The realization of these systems that connect a cybernetic component to the factors of the physical world presents immense challenges, and considering that their origin is relatively new compared to other areas of software, an increase in the complexity of the software developed and in all the tools and libraries in this field is of essential intent, assuming also, that more than ever the presence of failures or chaotic

behavior of these systems can lead to catastrophic consequences not only economically but also in the health of the individuals that surround the systems.

The notoriety, importance and characteristics of such systems can be seen in examples such as automatic derivation, aeronautical derivation, drones, power plants, pacemakers among many others all present in the various areas and sectors of our community and with prospects of increasing in the future years. Of these examples, special emphasis and importance is given to the various tasks in autonomous piloting, currently being addressed in a research project led by the (co)supervisors - which include following a leader, avoiding an obstacle, and landing or controlling the trajectory of a spacecraft.

Abstracting these cyber-physical systems into more mathematical and software orientated systems creates the opportunity for their analysis, development and verification using the right computational tools. However, there is still a general lack of tools based on these models, which arises from the fact that this is a relatively new field of study. Many of the few tools that already exist do not have well tested methodologies and techniques for a disciplined/systematic modeling and analysis. The tool Lince tries to combat this problem by offering an environment capable of interpreting hybrid programs based on a well-established semantics, and consequently promotes the development of these programs with confidence in their safety.

1.2 Objectives

The tool Lince is very recent and still needs a phase of evaluation, testing, and verification to clarify its advantages and drawbacks. This is precisely the goal of the dissertation. To this effect, a set of cyber-physical systems will be selectively chosen taking into account their importance in society, behavior, complexity and most prominent features. We will then present an analysis and their abstraction into a hybrid program. For each type of system, systematical techniques will be developed accounting to their future use in the development of other related hybrid programs.

The analysis of these systems will be performed via the tool Lince, and for each study conclusions will be drawn regarding its impact and challenges faced. In addition to this main objective, the fact that our exploration is systematic and based on a taxonomy of existing cyber-physical systems means that it has the potential to help delineate previously unknown methods and techniques in hybrid programming.

1.3 Report's structure

This report serves as introductory material to the final dissertation document that will be presented when the thesis is completed. Although certain portions of this document will be incorporated into the final document, it is incomplete with respect to the total scope of the project and therefore consists of only 3 parts.

- The first part, i.e. the current chapter, introduces the theme of the dissertation in a simple and motivating way, presenting as well its context and objectives.
- The second and following part will effectively be the state of the art, with the objective to present to the reader theoretical information and support about the concepts related to the project.
- The last and third part consists of conclusions and an explanation of the work still to be done, in particular the research methodology to be followed in the next stages of the dissertation and the planning of the tasks to be carried out

Part II

State Of The Art

Chapter 2

Concepts

2.1 Cyber-Physical Systems

Cyber-physical systems are described by their technicality of being composed of two components interconnected with each other, a physical component relative to the factors of the environment such as space, movement, time, among other processes and a cybernetic component relative to the processes of computation, communication and control. More simply, these systems can be explained by the following statement

> Cyber-physical systems combine cyber capabilities with physical capabilities to solve problems that neither party could solve alone.

Following this concept it may seem that cyber-physical systems involve, in one way or another, almost all, if not all, applications of technology in society, but it is important to note that this denomination is used mostly for those systems whose behavior is extremely interconnected to the physical processes to which they are subject. Robots are great examples, as they all physically move in space in a controlled and calculated way by discrete computational processes that adjust the various actuators (e.g. brakes) based on sensor readings made to the physical environment (space). Examples of **CPS** are not limited to these, its concept is so general that it expands into almost any field and industry that aims to perform complex tasks but in a safe and efficient manner, other example include: Automatic or assisted driving of automobiles and hive devices, calculating trajectories of aircraft and satellites, collision avoidance, train braking, cameras, power plants, medical devices, and even assisting surgeries for quick decision making.

Its importance is immense for the future of technology mainly for the comfort it can offer but also to help prevent catastrophes. This technology can for example prevent accidents by keeping a car always in its lane, taking control of the car if necessary, however, reaching its potential presents great challenges, in this case, the system would have to detect circumstances where trajectory correction is necessary and

distinguish with high confidence the cases in which the user wants to change lanes between accidental cases. Additionally, this system cannot only consider immediate results, but must also take into account future consequences caused by the trajectory changes made. Another example is the realization of an automatic driving system that makes one car able to follow another, avoiding collisions or drifting away, such quality could for example ensure the driving of two or more cars by a single driver, the leader. Again, challenges encountered are around the development of an algorithm capable of knowing when to accelerate, brake, or change trajectory by calculating such decisions extremely quickly and safely, not having time to plan all possible decisions and having to adapt to all scenarios. How can you prove that a developed algorithm actually guarantees all these demands, how can you certify it?

The study and development of cyber-physical systems involves a deep analysis between the two components cybernetic and physical but also the connection between them, how they communicate with each other and how one component influences the other. It is safe to say that the design of these systems is extremely complicated and needs strong foundations and strong tools to support all its possibilities.

2.1.1 Safety

As can be deduced from the fact that cyber-physical systems are built around dangerous environments where lives can be in danger if they malfunction, security is an important or the most important factor in their development, an invalid **CPS** can lead to the opposite of contributing to efficient and safe decision making leading to many catastrophes. Security is then the key to ensure the quality of a CPS, there is then a need to involve computer languages to enhance the development of these that take into account both the physical and cyber aspects of the system

To develop a completely fail-safe system would require a complete and correct understanding of the physical world and a complete catalog of all actions that can affect that world as well as a correct understanding of the consequences of those actions. While such a purpose is not currently achievable, the point is that the safety of a CPS and the validation of its objectives depends directly on how it is computationally programmed and how well it has been evaluated for errors. Assurances originating from analytical foundations in the form of computational languages are then necessary, and it is appropriate that these tools are adaptable and standardized to various or all situations to establish a firm foundation in the area of cyber-physical systems development, which the LINCE tool certainly tries to achieve Simple practical tests although still important, prove to be insufficient as they cannot cover all scenarios, which is extremely relevant knowing that the behavior of an CPS depends entirely on the circumstances in which it is situated. Without a mathematical generalization it would be impossible to offer a sufficiently qualified proof

2.2 Hybrid Systems

Although it is an easily understood description,in a study and programming context the definition of **CPS** is not relevant since little importance is given to the fact that a system is the participation of its cybernetic and physical parts. What really matters in a more specific context is to try to identify the characteristic behavior that all these systems exhibit. Taking that to account, it was concluded that in general CPS share a fundamental mathematical functioning and at it's core, cyber-physical systems are actually complex hybrid systems.

Hybrid systems are mathematical models of dynamical systems that combine discrete and continuous dynamics. Their behavior then includes both aspects that change discretely one step at a time and aspects that change continuously with continuous functions over time. In the previous example of automatic derivation following a leader the car continuously follows its trajectory over a time-derived function, However, the pilot, or in this case the computer, has the ability to change this trajectory or change the speed in discrete decisions to prevent collisions or departure from the leader. In this case, the car schedules a decision in a specific instant of time being a discrete dynamic, then the car follows the result of that decision continuously for a period of time until it finds itself in a new situation that leads to re-evaluate a new decision to be made. Other examples of hybrid systems can usually be found in the areas of control theory, event-based mechanics, software incorporation, and biology, including for example disease propagation and personalized cancer treatments.

During the conversion from cyber-physical to hybrid systems, despite initial deductions, processes often implemented in a continuous dynamics do not necessarily imply representation of a physical process, and processes implemented in a discrete dynamics are not necessarily a cybernetic component. For example, some computational processes are computed so quickly and frequently that it is more advantageous to interpret them with a continuous dynamics even if it is not completely true. In fact, this is one of the advantages of hybrid systems, their mathematical principles are so basic and fundamental that they allow us to interpret processes either continuously or discretely, offering a trade-off depending on the goal we want to achieve. For some purposes, it may be better to model the landing of an aircraft as a discrete change of state from air to ground. For other purposes, such as developing an autopilot program for landing that same spacecraft, it is important to include a much more refined view, because the aircraft will simply take off again if it has a high speed.

In conclusion, hybrid systems are distinguished from cyber-physical systems in that the latter are mathematical models of complex systems involving discrete and continuous dynamics, while cyber-physical

systems are described mostly by their technical characteristics, however, despite their differences, hybrid systems can often represent behaviors of a cyber-physical system, and are therefore used as abstractions of CPS.

However, it is important to note that certain biological processes such as genetic networks can be captured as hybrid models despite not being cyber-physical systems. Contrarily, certain cyber-physical systems may exhibit additional features beyond which hybrid systems are capable of representing. Of these additional features, a cyber-physical system may for example exhibit random or uncertain behavior derived from choices influenced by variability in the environment, exhibiting a so-called stochastic dynamics if good distribution of available choices is known. Other dynamics such as non-deterministic, adversarial or distributed may also be necessary to represent a CPS, however, Lince does not currently contain foundations for the development of programs with these dynamics, due to the difficulty and complexity of formalizing the necessary logical and semantic composition between the different aspects, and these cases will not be addressed in this dissertation. Nonetheless, these topics can be mentioned and explained in the future as possible improvements for the tool.

2.3 Hybrid Programming

Hybrid programming is an emerging computational paradigm that aims to use principles and techniques from programming theory to develop computational components that interact closely with physical processes. The distinction between a hybrid system and a hybrid program is emphasized: A hybrid system are those systems whose behavior has both discrete and continuous characteristics. Hybrid programs are specifications of hybrid systems written in a programming-oriented style. An easy way to recognize the distinctions between the two is to keep in mind that a single hybrid system system can have several different programs developed to represent it.

The following example of a hybrid program is observed:

```
while true do  if v ≤ 10 then (v' = 1 for 1) else (v' = −1 for 1)
```

This example represents a digital motion controller (Cruise Control) that periodically measures and regulates the speed of a vehicle (v): if it is less than or equal to 10, the controller accelerates for 1 unit of time, as dictated by the program instruction v' = 1 for 1 (v' = 1 is a differential equation representing the rate of change of velocity per time). Otherwise, it decelerates for the same period of time (v' = -1 for 1).

Notice that, unlike normal languages, this speed controller does not only contain normal constructs like *while-loops* and *conditions* but also differential constructs, so it is similar to the kind of programming that many engineers are used to but extends to new features. The use of differential equations is a characteristic feature of hybrid programs that allows the representation in programming language of various physical processes mentioned earlier, namely various components such as time, energy, velocity and motion. While the *While-language* used as a basis can easily represent the discrete behaviors of hybrid systems.

There are however other formalizations of hybrid systems namely hybrid automata, an extension of the classical automaton with state variables that evolve continuously while in a state and change discretely at state transitions. Hybrid automata have a rich theory and powerful tools, both typically focused on reachability analysis unlike hybrid programs.

Chapter 3

Tools for Modeling and Analysis of Cyber-Physical Systems

The immersion of cyber-physical systems led, throughout the years, to the creation of various powerful software tools capable of helping in their development, analyses and verification. Because of the complexity of dynamic systems there isn't a right or better tool, but rather the appropriate tool to a given problem and objective, it's then important to highlight why exactly the creation of Lince was beneficial to the field of study by giving a short introduction to some of the various tools available on the market, how they work, their characteristics and uses. These short explanations help understand the current state of the field and how despite all the progress made, there is still a necessity to keep innovating in order to achieve all the potential of cyber-physical systems.

3.1 Uppaal

Just like all the tools in the field Uppaal has the objective to help develop dynamic systems. The difference present is in how the tool allows the representation of these systems, and giving the available abstraction what dynamic components of a cyber-system can be integrated. The structure in cause is a web of timed automaton extending to the usage of different language data types (strings, integers), clocks and operations between those. Each timed automata represents a process and consists of locations or nodes and transitions or edges, each node and edge can be paired with a set of invariants being then capable of representing certain conditions that make a state or transition possible and therefore portray the behavior of the system.

Timed automaton allow us to represent various different dynamics previously mentioned. In a location, the system have a continuous dynamic represented by the flow of time in the form of clocks, in an edge it's possible to update variables and clocks, altering the state of the system and representing a discrete

dynamic. The usage of channels and shared variables allow us to synchronize transitions of different automata being one of the few tools to easily be able to study distributed systems. In addition, a timed automata contains a non-deterministic behavior meaning that, the simulation of a system can lead to different results or paths depending on the different possible transitions taken in that simulation. Because of the usage of timed automaton, Uppaal is built using at it's core a pseudo-formal semantics, allowing to logically determinate the valid actions of a model in a certain state, being delay transitions made to advance clocks or action transitions.

Although a bit slow, by taking in to account the simulation of all possible depth limited paths, Uppaal makes available a powerful verification query language capable of expressing different type of properties:

- **Reachability**. A specific condition holds in some state of the model potential behaviors.
- **Safety**. A specific condition holds in all the states of an execution path.
- **Liveness**. A specific condition is guaranteed to hold eventually.
- **Deadlock**. A deadlock is possible or not in the model.

One very simple example, just for demonstration purposes, of a system in Uppaal is a Light Switch where the user is able to press a button two consecutive times in less than a specific time to turn the bright light and only one to turn the low light. A distributed system with two timed automata, the lamp and the user.

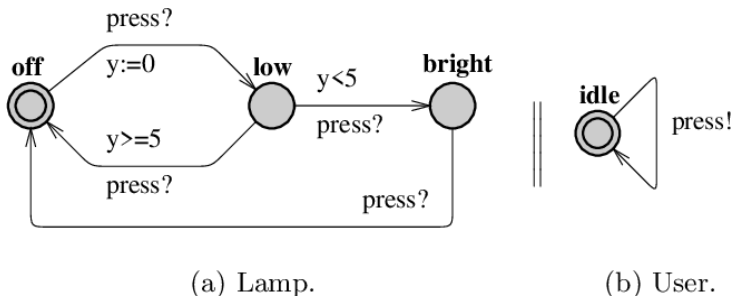


Figure 1: Light Switch in Uppaal

In conclusion, Uppaal is a very powerful tool, especially to specify time based distributed systems, for example, entities competing for resources, but its graphic interface is not very flexible and can be become very complex very easily. The continuous dynamic available is restricted to clocks, not being able to represent for example space (trajectory and velocity) or energy based cyber-physical systems.

3.2 Simulink

Simulink is an extra component of Matlab that allows rapid and interactive graphical construction of virtual block orientated models that simulate and analyze dynamic systems. The objective is to be able to produce a running model that would otherwise require hours to build in the laboratory environment. These models work by the directional and continuous or sampled flow of numeric/signal values, originating from one or more source blocks and terminating in final blocks, being re-directed, altered or filtered in the blocks in-between.

These models are built recurring only to predefined blocks categorized in:

- **Sources**: Used to generate various signals. Ex:Constant input,Sine Wave, Step, etc.
- **Sinks**: Used to output or display signals. Ex:Scope, Display, To Workspace, XY Graph, etc.
- **Discrete**: Linear, discrete-time system elements. Ex: Discrete Filter, Discrete State-Space, Discrete Transfer Fcn, Discrete Zero-Pole, Unit Delay, etc.
- **Continuous**: Continuous-time system elements. Ex:Integrator, State-Space, Transfer Fcn, etc.
- **Signal Routing**: Contains useful blocks to build a system. Ex: Mux, Demux, Switch, etc.
- **Math Operations**: Contains common math operations. Ex:Abs, Gain, Product, Sign, Sum, etc.

The next simple example, as we can see, serves to represent a speaker. A sinusoidal wave (sound) is produced in a source, multiplied by the math operation block Gain, and both the original and the altered sound are displayed in a sink Scope.

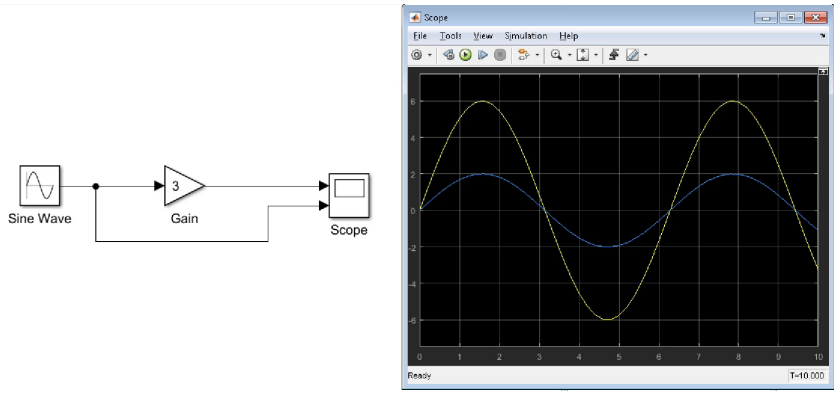


Figure 2: Simple Speaker example in Simulink

Simulink is a useful and easy tool, but it represents certain disadvantages derived from the fact of involving graphical, interactive and predefined component design, although the reuse of models is possible, it's not very scalable or flexible representing several difficulties for example in abstracting cyber-physical systems involving three dimensional movement, something essential in the field.

3.3 KeYmaera X

KeYmaera makes available a very lightweight environment capable of analyze **Hybrid Program**, being one of the few examples where the analyses of these programs is not made by simulation but by proof deduction, recurring to the use of a well defined and established differential dynamic logic (dL), a logic for specifying and verifying properties of hybrid systems with mixed discrete and continuous dynamics. The major advantage of KeYmaera is the ability to use aggressive proof search to verify with confidence certain safety conditions.

One example of a hybrid program in KeYmaera and its corresponding verification is the bouncing ball, a system that tracks the movements of a ball jumping into the ground. After defining the assumptions and behavior of the program, it's then possible, using this tool to verify for example the invariant related to the conservation of mechanical energy when the damping coefficient equals to 1.

```
ArchiveEntry "LFCPS Textbook/07: Bouncing Ball"
Description "7.4: Acrophobic Bouncing Ball".

Definitions      /* function symbols cannot change their value */
  Real H;        /* initial height */
  Real g;        /* gravity */
  Real c;        /* damping coefficient */
End.

ProgramVariables /* program variables may change their value over time */
  Real x, v;     /* height and vertical velocity */
End.

Problem
  (x>=0 & x=H & v=0) &
  (g>0 & 1=c&c>=0)
 ->
  [
    {
      {x'=v,v'=-g&x>=0}
      {?x=0; v:=-c*v;  ++  ?x!=0;}
    }* @invariant(2*g*x=2*g*H-v^2 & x>=0)
  ] (x>=0 & x<=H)
End.
```

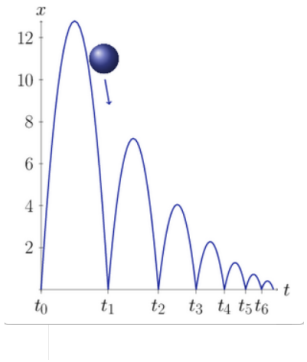


Figure 3: Bouncing Ball Verification in KeYmaera X

Because there is no simulation involved, the tool is useful to verify predefined hybrid programs but doesn't help during the development of those, not being able to study the course of the system and making changes accordingly.

3.4 Modelica

Modelica is a software language used for modeling **Cyber-physical System (CPS)**, the abstraction of these systems is made by connecting different components that represent physical objects like turbines or pumps, being these reused or created using mathematical equations to represent a set of rules that define it's principals.

What makes this tool so distinguished from the others in the field is that the language is acausal meaning the components are described by relationships between system variables, there is no clear sequence or procedure of actions, and the tool will sort and compile this relationships into execution code to suit the system. This presents a grand advantage in the development of more practical and physical cyber-systems like electric, magnetic, hydraulic or thermal systems being used a lot in automotive companies to design energy efficient vehicles. Modelica is a testament to the scope of cyber-physical systems and their possible mathematical model abstractions.

3.5 Lince

Lince is a newly created tool with the objective of combating the difficulties found today in the hybrid programming environment. This tool,that counted with the presence of the co-supervisors in its development, functions as a hybrid program interpreter and simulator originating from the extension of the use of differential equations to a classical *while-language*, thus promoting a composed vision that remains in touch with classical programming theories. This approach differs from various other hybrid system development and analysis programs on the market precisely because of this characteristic of being orientated to a programming style, as opposed to the usual construction of graphical diagrams, therefore being more adaptable, flexible and more easily integrated with other languages or tools.

One of the desired goals is that in the future Lince can be used as a complement to other debugging tools, especially those that work in the hybrid domain. However, the main feature and advantage of the tool is that it has an extremely well defined, established and formalized semantic being therefore built on logically deduced and theoretical foundations taking into account the composition of its different components (*while-language* and *differential*) and the desired functionalities. In fact, one of the main concerns during its development was to make sure that the interpretation of hybrid programs at the implementation level coincides precisely with the intended semantics calculated that integrates both the *while-language* and its continuous behavior in a rigorous and safe way, however, the demonstration and explanation of

this semantics are not within the scope of this dissertation.

Other tools mentioned before like *SIMULINK* and *MODELICA* that are standard in the hybrid systems analysis and development industry lack these formalized semantics, even though, there have been efforts to provide these for subsets of both tools, so that simulation errors caused by numerical approximation methods are avoided. It is also important to note that, other advantage in comparison with other tools is that in the future, by extending its semantics it's possible to take into account the addition of other physical behaviors such as randomness. In spite of all that, its recent creation carries various disadvantages to the use of the tool, the lack of testing and well defined techniques paired with its core basic function and the absence of well defined purposes can prevent its application in real life problems, something that this projects intends to address.

In summary, Lince acts as a practical framework to formally specify, analyze and simulate hybrid programs and present relevant properties regarding its behavior, such as the cruise controller previously shown:

while true do if $v \leq 10$ then ($v' = 1$ for 1) else ($v' = -1$ for 1)

In this example in particular, Lince has the capacity to safely answer questions such as: What is the maximum speed the vehicle will reach? Does it ever reach an unsafe speed? And if so, for how long?

As can be demonstrated by the following image that represents the result of running the cruise control in the tool,in this context Lince is, in a very specific way, a tool that given a hybrid program and an instant of time t, returns the value to which the program evaluates for time t.

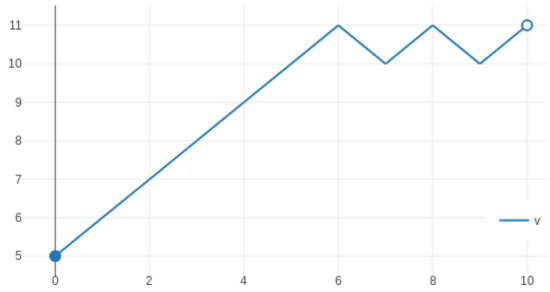


Figure 4: Velocity Variation Example in Lince

Lince offers a lighter verification method that can be complementary to a heavier verification tool like KeYmaera, considered then to be a basic building block not only for developing programming tools in the hybrid scenario, but also for simulating hybrid system executions, believed to be ,at best of our knowledge, the first tool of its kind to be created.

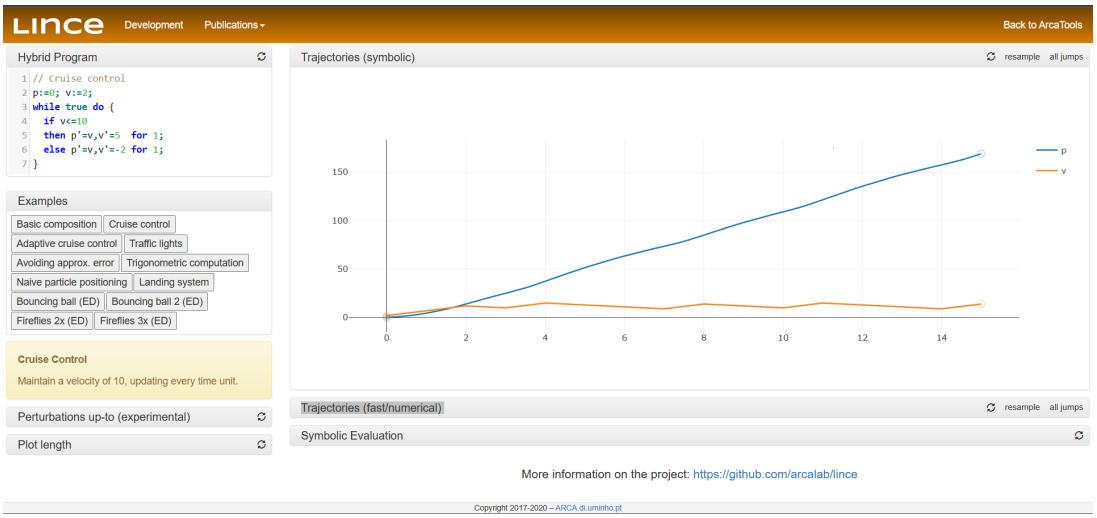


Figure 5: Online Lince Example

Part III

Conclusion

Chapter 4

Conclusion

In conclusion, this **PTR** is intended to introduce the topic and purpose of the dissertation to the reader, the state of the art in particular should give the reader knowledge about the topics essential for understanding the importance and methodology of the future work to be done. Briefly, we explored that cyber-physical systems represent an essential field in the future of technology for the further integration of technology with the physical world leading to automation and/or assistance of several important tasks in an effective, safe, and easier manner. We described the characteristics of hybrid systems and programs and how they represent reliable abstractions of **CPS** capable of assisting in their development and analysis. We explained the current difficulties that such field of study brings namely the lack of strategies, methodologies, tests and tools thus introducing the recent existence of Lince, its base capabilities and its distinctive features in the market.

In the future, various hybrid programs will be developed using Lince, recurring to the development of systematical techniques that will serve as foundations to leverage the future utilization of the tool in real-life situations. This study will also provide the opportunity to address particular special situations where the usage of the tool is advantageous, and in inverse obstacles that the tool presented and should exceed in future versions.

4.1 Investigation Methodology

It is important before working on any solution to plan the work to be performed. As mentioned earlier, this dissertation will involve the individual study of several cyber-physical systems and their consequent hybrid programs in the tool Lince, such that the following diagram clarifies the approach to be taken for each case study and what tasks it consists of.

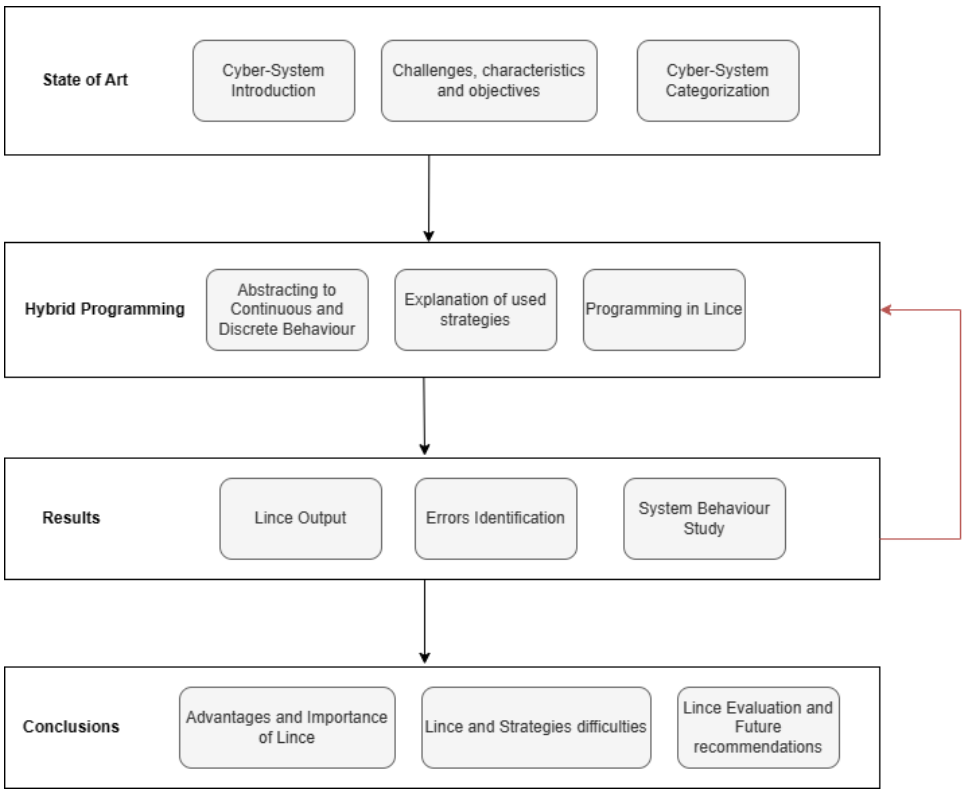


Figure 6: Investigation Methodology

4.2 Planned Schedule

The planned schedule to be followed during the course of this dissertation is also presented, containing the different tasks to be carried out until the final delivery of the thesis.

Tarefa	Nov	Dez	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago
State of the Art	●	●	●							
Preparation of **PTR**			●							
Study of **Hybrid Program**				●	●					
Categorization of **Hybrid Program**				●	●					
Modeling and Analysis of **Hybrid Program**					●	●	●	●		
Dissertation Writing									●	●

Table 1: Schedule.

Bibliography

Peter Fritzson. *Principles of Object Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach*. John Wiley & Sons, 2014.

Nathan Fulton, Stefan Mitsch, Jan-David Quesel, Marcus Völp, and André Platzer. *KeYmaera X: An Axiomatic Tactical Theorem Prover for Hybrid Systems*. In Amy Felty and Aart Middeldorp,, 2015.

Sergey Goncharov, Renato Neves, and José Proença. *Lince: Lightweight Prototyping of Hybrid Programs*.

Sergey Goncharov, Renato Neves, and Jose Proençaa. *Implementing hybrid semantics: From functional to imperative In International Colloquium on Theoretical Aspects of Computing*, pages 262–282. Springer, 2020.

Harold Klee and Randal Allen. *Simulation of dynamic systems with MATLAB and Simulink.* CRC Press, 2007.

Thomas Krilavicius. Bestiarium of hybrid systems. 2005.

Thomas Krilavicius. Hybrid Techniques for Hybrid Systems. PhD thesis, University of Twente. 2006.

Renato Neves. Hybrid programss. PhD thesis, Minho University. 2018.

Andre Platzer. Keymaera x tutorial, a. URL https://keymaerax.org/Xtutorial.html.

André Platzer. *Logical Foundations of Cyber Physical Systems*. Springer Nature Switzerland, b.