UNIVERSITÀ DEGLI STUDI
DI TRENTO

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE
ICT International Doctoral School

An SMT-based framework for the formal analysis of Switched Multi-Domain Kirchhoff Networks

Mirko Sessa

Advisor

Prof. Alessandro Cimatti

Fondazione Bruno Kessler - Trento - IT

Co-Advisor

Prof. Sergio Mover

École Polytechnique - Paris - FR

October 2019

In memory of my friend Alessia Gadotti (1987-2017)
and of her irrepressible joy for living.

To my twin soul Lia
for her uncomplaining and invaluable support.

Abstract

*Many critical systems are based on the combination of components from different physical domains (e.g. mechanical, electrical, hydraulic), and are mathematically modeled as Switched Multi-Domain Kirchhoff Networks (*SMDKN*). In this thesis, we tackle a major obstacle to formal verification of* SMDKN*, namely devising a global model amenable to verification in the form of a Hybrid Automaton. This requires the combination of the local dynamics of the components, expressed as Differential Algebraic Equations, according to Kirchhoff's laws, depending on the (exponentially many) operation modes of the network.*

We propose an automated SMT-based method to analyze networks from multiple physical domains, detecting which modes induce invalid (i.e. inconsistent) constraints, and to produce a Hybrid Automaton model that accurately describes, in terms of Ordinary Differential Equations, the system evolution in the valid modes, catching also the possible non-deterministic behaviors. The experimental evaluation demonstrates that the proposed approach allows several complex multi-domain systems to be formally analyzed and model checked against various system requirements.

Keywords

Switched multi-domain Kirchhoff networks, differential-algebraic equations, hybrid automata, model checking, satisfiability modulo theories

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 General context

Complex critical systems are often formed by the interaction of components from multiple physical domains (e.g. electrical, hydraulic, and mechanical). An example from aerospace is a landing gear system [BW14],

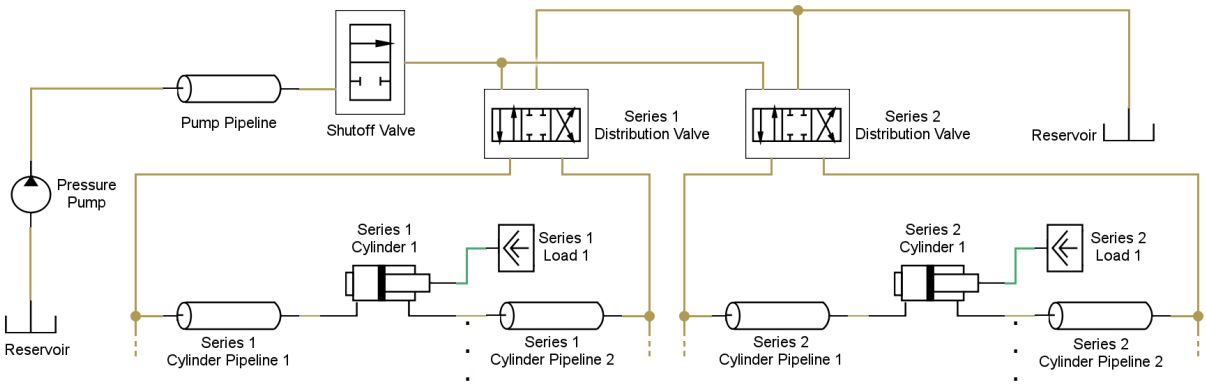


Figure 1.1: Landing Gear System with $N = 2$ hydraulic cylinder lines ($\text{LGS}_{[N]}$).

depicted in Figure 1.1, where the pressure applied by a hydraulic circuit (including pumps and valve) operates moving components from the hydro-mechanical domain (e.g. a cylinder) that in turn drive mechanical loads. A key aspect of these systems is that the basic components experience a multidirectional interactions due to the bidirectional propagation of energy in the network (e.g. a short-circuit propagates bidirectionally in an

electrical network). Moreover, the basic components might react with non-deterministic time delay to external events. The modeling formalism must consider these aspects. Basic components (e.g. valves, and cylinders) have multiple operation modes and exhibit hybrid dynamics. These dynamics include continuous behaviors, typically described by Differential-Algebraic Equations (DAE) associated to the modes, and instantaneous changes (or switches) among modes. The connection of basic components into composite systems is often modeled as Switched Multi-Domain Kirchhoff Networks (SMDKN) [Jan11]. Each combination of the components modes determines a (global) mode of the network. For each global mode, the continuous dynamics is represented by the system of DAE obtained by joining the equations that characterize each component in the respective mode with the equations that correspond to the Kirchhoff's connection laws.

1.2 Challenges

In this thesis, we investigate methods for the formal analysis of SMDKN, tackling two key challenges. The first challenge is to convert a DAE-based network description into a formalism based on Ordinary Differential Equations (ODE) and that is amenable to formal verification. The existing formal verification tools for hybrid systems [FGD+11, GKC13a, CGMT15] take as input *hybrid automata* and, in most cases, require a description of the continuous dynamics in the form of ODE. Obtaining an ODE from a DAE is possible with a process called *reformulation* [Ria08]. One could thus conceive an approach that iterates over the network modes, reformulates for each of them the corresponding DAE into an ODE, and recombines the resulting ODE into an automaton. Unfortunately, this iterative approach is unfeasible in practice: the number of modes of a switched network is exponential in the number of components.

The second challenge stems from the fact that the reformulation cannot always map a DAE onto an ODE. In fact, a DAE is a relational characterization deriving from a constraint-based description of the reality, while an ODE is in essence a functional description. Thus, under certain conditions, a DAE may be inconsistent (i.e. infeasible from the physical standpoint) or under-constrained (i.e. some physical quantities are undetermined). Unfortunately, inconsistencies and under-specifications may be hidden in the (exponentially many) modes of the network, and may be hard to spot.

1.3 Contributions

In this thesis, we propose a general method to reformulate SMDKN into hybrid automata with ODE dynamics. In order to deal with multi-domain networks, we propose a purely algebraic, general argument, which guarantees the existence of the reformulation, generalizing the *Implicit Function Theorem* [Mun97] for linear systems. The method is able to synthesize the modes free from inconsistencies and under-specifications, and to present them in the form of diagnostic information.

1.4 Technical approach

We adopt an approach based on Satisfiability Modulo Theories (SMT) [BSST09] to reason about the algebraic representation of DAE-based networks. We build on the ability of modern SMT solvers to carry out quantifier elimination and to deal with huge sets of assignments to discrete variables. We exploit the algebraic nature of the problem, in particular the linearity principle holding for the DAE associated to each network modes, to aggressively simplify the expensive quantifier elimination steps.

1.5 Evaluation

We perform an experimental evaluation on several multi-domain scalable real-world benchmark applications and on a railway case study developed in collaboration with the Italian train company [CCM+18]. The proposed optimizations substantially increase the scalability of the procedures, allowing us to validate and reformulate SMDKN featuring millions of modes. We verify the hybrid automata resulting from our procedures by means of some existing SMT-based verification tools (e.g. HYCOMP [CGMT15]).

1.6 Structure of the thesis

The thesis is organized as follows. In Chapter 2 we work out a complete example to overview the key aspects of the entire work. In Chapter 3 we present the mathematical background. In Chapter 4 we define the syntax and semantics for Kirchhoff networks. Chapter 5 defines the validation and reformulation problems for SMDKN. In Chapter 6 we first present a baseline symbolic solution, and then we describe the optimized symbolic approach that exploits the algebraic structure of the problem. Chapter 7 presents the experimental evaluation. and Chapter 8 discusses the railway case-study. Chapter 9 discusses the related work. In Chapter 10 we draw some conclusions and discuss the directions for future work.

1.7 List of publications

The work presented in this thesis appeared in three main conference papers at FM16, FMCAD17, and FMCAD18.

At the conference FM16, the paper [CMS16] presented a method to convert Switched *Electrical* Kirchhoff Networks into hybrid automata. The work was limited to electrical networks because the SMT encodings of the

conversion was built on well-known results from graph-theory on the graph representation of the network. Additionally, non-deterministic behaviors of the algebraic variables were not allowed.

At the conference FMCAD17, the work [CMS17] proposed a more general validation and reformulation approach than [CMS16] in two respects. First, we were able to deal with *multi-domain* networks, enabling mechanical, electrical and hydraulic domains, and their combination, whilst [CMS16] was restricted to electrical networks. Second, the method in [CMS16] were only able to produce a hybrid automaton if the electrical network fulfills the conditions of existence and determinism in *all the modes* and for *all the variables*, while [CMS17] analyze SMDKN with *non-deterministic* algebraic variables as well. Both extensions are made possible by the adoption of a theoretical settings that is significantly more general than the domain-specific topological approach on the network graph used in [CMS16].

Finally, the work [CCM+18], appeared at FMCAD18, presented the application of our SMT-based modeling and analysis approach to a railway case study on Relay Interlocking System developed in collaboration with the Italian train company. We experimented an approach based on this work to understand legacy relay circuits in the railway domain. We rely on an accurate representation at the physical level in form of Switched Kirchhoff Networks, that is then reduced to a symbolically represented network of hybrid automata, and then analyzed by means of SMT-based model checking. The experimental evaluation demonstrated the precision and scalability of the analyses. The proposed methodology is at the core of an ongoing research project aiming at the in-the-large analysis of legacy railway interlocking and the open specification of computer-based solutions. To the best of our knowledge, no other works address the verification problem of a Relay Interlocking System based on its hybrid physical behavior.

Chapter 2

Overview of the approach

In this chapter, we progressively work out a full network in order to provide an overview of the basic concepts behind this work. Our goal is to intuitively show the main challenges encountered to formally model-check a switched Kirchhoff network with the existing tools, and to present the key aspects of the proposed SMT-based approach. We believe that presenting a simple electrical network will help to clarify the challenges and the problems solved by our approach. We start from the description of the electrical network, of the component interconnections and of the modeling approach of the components. Then we describe the issues coming from ill-posed network configurations encountered in the reformulation process from SMDKN to a hybrid automaton with ODEs amenable to symbolic model checking.

The *Switched Multi-Domain Kirchhoff Network* (SMDKN) formalism describes mechatronic systems that mix electrical, hydraulic, mechanical, and thermal switching components. Figure 2.1 shows the diagram of the SMDKN of an electrical circuit composed of a current source (I_S), two resistors (R_0 and R_S), three switches (S_0, S_1 and S_2), and two capacitors (C_1 and C_2). The positive and negative terminal of each component are graphically denoted by a plus (+) and a minus (−) sign, respectively. The

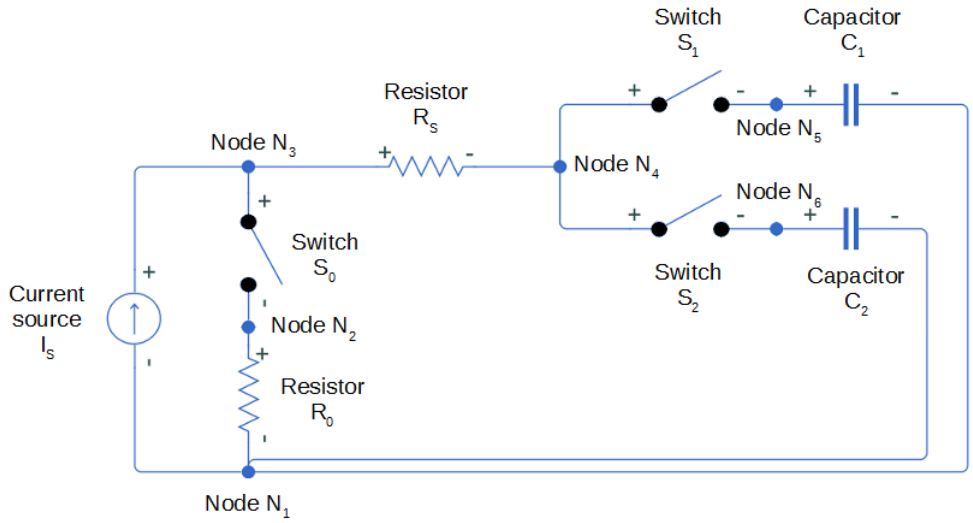


Figure 2.1: Switched electrical Kirchhoff network of the battery charging system represented in the M_1 discrete configuration where the switches S_0, S_1 and S_2 are open. The connection nodes between component terminals are denoted by blue circles. The positive and negative terminals of the components are denoted by a + and - sign.)

interconnections of the component terminals form six connection nodes (N_1, N_2, N_3, N_4, N_5 and N_6) graphically denoted by the blue circles.

The network models a simple two-phase *battery charging system* where the current source charges the two batteries that are modeled as two capacitors. The first charging phase, characterized by a battery voltage lower than a design threshold, must be performed with a constant-rate current; the second charging phase, where the battery voltage is between the threshold and the maximum voltage, must be performed with a constant-rate voltage to prevent battery damages. The batteries must be charged one at a time because they cannot be connected in parallel due to possible overcurrents.

The switching behavior of the network depends on the three ideal switches: every switch has an *open* and *closed* mode, consequently the cross-product of the three switches modes returns the eight global modes of Table 2.1. In general, the number of global modes of the network increases

exponentially with the number of switching components in the network. Our SMT-based reasoning approach deals with this exponential complexity of the problem to make the proposed solution feasible for real-world application.

We assume that the switches are externally operated by a controller (omitted in Figure 2.1) that, if properly designed, fulfills the specification described above. The goal of the designer is to verify whether the composition of the network with the controller satisfies the system requirements.

Mode name	Switch configurations
M_1	S_0 open, S_1 open, S_2 open
M_2	S_0 open, S_1 open, S_2 closed
M_3	S_0 open, S_1 closed, S_2 open
M_4	S_0 open, S_1 closed, S_2 closed
M_5	S_0 closed, S_1 open, S_2 open
M_6	S_0 closed, S_1 open, S_2 closed
M_7	S_0 closed, S_1 closed, S_2 open
M_8	S_0 closed, S_1 closed, S_2 closed

Table 2.1: The eight discrete configurations of the network of Figure 2.1

2.1 Modeling of the interconnections

A component can be considered as a box that connects with the other components in the network via a set of *terminals*. A terminal represents a physical interface of the component that affects two physical quantities, called in general the *effort* and the *flow*. Effort and flow generalize the physical quantities exchanged between components in mechatronic systems. In every physical domain the effort and flow take a particular dimension (see Table A.1 for more details). For example, in the electrical domain the effort corresponds to the *electrical potential* on the terminal while the flow corresponds to the *electrical current* flowing through a terminal. As reference

direction, the flow is considered positive when it goes from the terminal to the internal of its component. All the components of the network of Figure 2.1 are two-terminal components of the form shown in Figure 2.2.

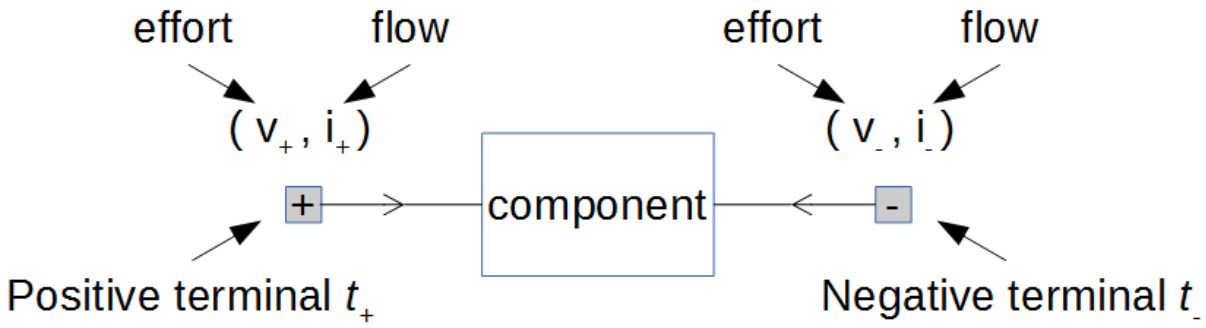


Figure 2.2: Conceptual model of a two-terminal component. The positive reference direction of the flow variables is depicted with an empty arrow going from the terminal to the component.

Multiple terminals are connected together forming a *node* of the network, and the terminals involved in a node are subject to the Kirchhoff's connection laws. The two Kirchhoff's laws state that the potential on each terminal connected to the same node must be the same, and that the algebraic sum of the terminal currents entering the node must be zero. Figure 2.3 shows the generalization of the Kirchhoff's connection laws to the case of n terminals connected together. In Figure 2.1, for example, the negative terminal of the switch S_1 and the positive terminal of the capacitor C_1 are connected together forming the node N_5. This connection forces the potential on the two terminals to be the same, and the algebraic sum of the terminal currents entering the node to be zero. Table 2.2 reports the connection constraints for the six nodes of the network in Figure 2.1.

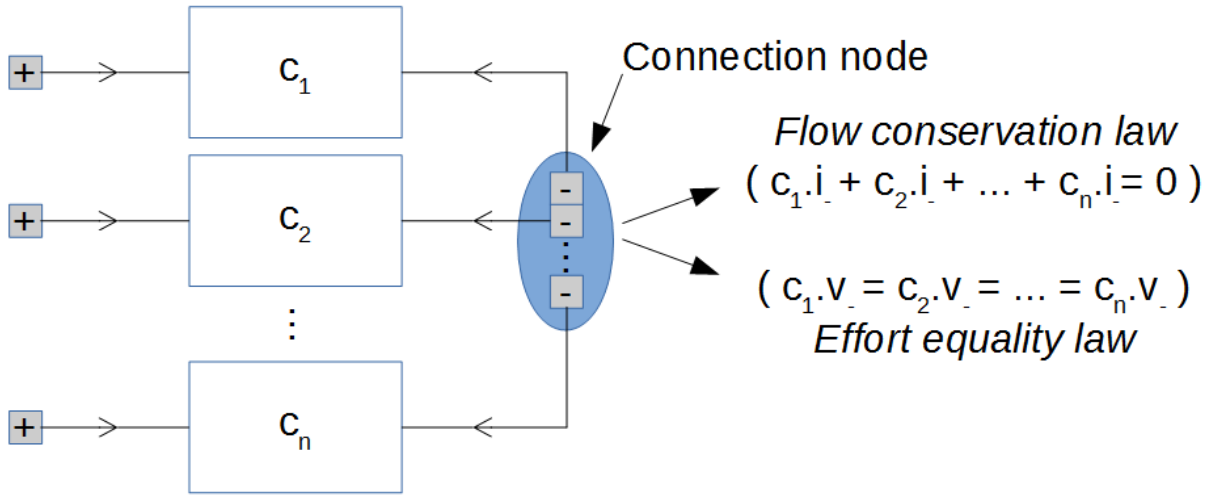


Figure 2.3: Kirchhoff's conservation laws on the connection of n terminals. We use the dotted-notation to refer to the effort and flow variables of a terminal: for instance, the effort variable v_- of the negative terminal of the component c_1 is denoted with $c_1.v_-$.

	Constraint	Explanation
N_1	$I_S.i_- + R_0.i_- + C_1.i_- + C_2.i_- = 0$	Conservation of currents law
	$I_S.v_- = R_0.v_-$	Equality of potentials law
	$I_S.v_- = C_1.v_-$	Equality of potentials law
	$I_S.v_- = C_2.v_-$	Equality of potentials law
N_2	$R_0.i_+ + S_0.i_- = 0$	Conservation of currents law
	$R_0.v_+ = S_0.v_-$	Equality of potentials law
N_3	$I_S.i_+ + S_0.i_+ + R_S.i_+ = 0$	Conservation of currents law
	$I_S.v_+ = S_0.v_+$	Equality of potentials law
	$I_S.v_+ = R_S.v_+$	Equality of potentials law
N_4	$R_S.i_- + S_1.i_+ + S_2.i_+ = 0$	Conservation of currents law
	$R_S.v_- = S_1.v_+$	Equality of potentials law
	$R_S.v_- = S_2.v_+$	Equality of potentials law
N_5	$S_1.i_- + C_1.i_+ = 0$	Conservation of currents law
	$S_1.v_- = C_1.v_+$	Equality of potentials law
N_6	$S_2.i_- + C_2.i_+ = 0$	Conservation of currents law
	$S_2.v_- = C_2.v_+$	Equality of potentials law

Table 2.2: Explanation of the connection constraints in the switching differential-algebraic equations of the network of Figure 2.1. All the constraints are algebraic and non-switching (i.e. always valid).

	Constraint	Type	Switching	Explanation
I_S	$I_S.i_- + I_S.i_+ = 0$	algebraic	No	Conservation of current
	$I_S.i_- = i_s$	algebraic	No	Generated current equal to i_S
R_0	$R_0.i_- + R_0.i_+ = 0$	algebraic	No	Conservation of current
	$R_0.v_+ - R_0.v_- = V_{R_0}$	algebraic	No	Voltage between the terminals
	$V_{R_0} = r_0 * R_0.i_+$	algebraic	No	Ohm's law with resistance r_0
R_S	$R_S.i_- + R_S.i_+ = 0$	algebraic	No	Conservation of current
	$R_S.v_+ - R_S.v_- = V_{R_S}$	algebraic	No	Voltage between the terminals
	$V_{R_S} = r_S * R_S.i_+$	algebraic	No	Ohm's law with resistance r_S
S_0	$S_0.i_- + S_0.i_+ = 0$	algebraic	No	Conservation of current
	$S_0.v_+ - S_0.v_- = 0$	algebraic	Yes	If S_0 is closed, short-circuit law
	$S_0.i_+ = 0$	algebraic	Yes	If S_0 is open, open-circuit law
S_1	$S_1.i_- + S_1.i_+ = 0$	algebraic	No	Conservation of current
	$S_1.v_+ - S_1.v_- = 0$	algebraic	Yes	If S_1 is closed, short-circuit law
	$S_1.i_+ = 0$	algebraic	Yes	If S_1 is open, open-circuit law
S_2	$S_2.i_- + S_2.i_+ = 0$	algebraic	No	Conservation of current
	$S_2.v_+ - S_2.v_- = 0$	algebraic	Yes	If S_2 is closed, short-circuit law
	$S_2.i_+ = 0$	algebraic	Yes	If S_2 is open, open-circuit law
C_1	$C_1.i_- + C_1.i_+ = 0$	algebraic	No	Conservation of current
	$C_1.v_+ - C_1.v_- = V_{C_1}$	algebraic	No	Voltage between the terminals
	$c_1 * \frac{dV_{C_1}}{dt} = C_1.i_+$	differential	No	Capacitor charging law
C_2	$C_2.i_- + C_2.i_+ = 0$	algebraic	No	Conservation of current
	$C_2.v_+ - C_2.v_- = V_{C_2}$	algebraic	No	Voltage between the terminals
	$c_2 * \frac{dV_{C_2}}{dt} = C_2.i_+$	differential	No	Capacitor charging law

Table 2.3: Explanation of the constraints of the components that form the switching differential-algebraic equations of the network of Figure 2.1. The resistances r_0 and r_S, the capacitances c_1 and c_2, and the generated input current i_S are real-valued constant parameters fixed at design time. The value of the current variable $I_S.i_-$ is externally determined by the parameter i_S and drives the evolution of the network, thus $I_S.i_-$ is the *input* real variable of the network. The values of the voltage variables V_{C_1} and V_{C_2} depends on the past-evolution of the network, thus V_{C_1} and V_{C_2} are the *state* real variables of the network.

2.2 Modeling of components

Every component of the network is a box that contains the mathematical description of its physical behavior in terms of *basic principles of physics*. Table 2.3 shows all the constraints of the components deployed in Figure 2.1, while Chapter A provides further details on the components used in this thesis.

2.2.1 Algebraic components

The electrical *current source* of Figure 2.4 is a two-terminal component that forces the current i_- on its network branch to be identically equal to a continuous function of time provided by the designer as a parameter of the component. In this thesis, for the sake of clarity, we consider mainly constant functions of time, but the proposed approach works for general continuous functions. Since the current i_- drives the network with a user-defined time-evolution, we say that i_- is an *input* variable of the network. An additional constraint $i_+ + i_- = 0$ describes the *current conservation law* through the component: the current entering one terminal equals the current exiting the other terminal. No equations about the potential variables are defined because the voltage drop of the component will be determined by the mutual-interaction of the component with the network.

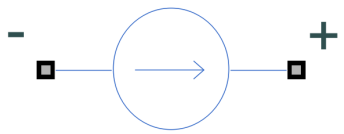


Figure 2.4: Electrical current source.

The electrical *resistor* shown in Figure 2.5 follows the *Ohm's law* $v_+ - v_- = r * i_+$: the voltage drop $v_+ - v_-$ between the terminals is directly proportional to the current i_+ through the component with a positive pro-

portionality constant r called resistance. Similarly to the current source component, the additional constraint $i_+ + i_- = 0$ describes the *current conservation law* through the component.

Figure 2.5: Electrical resistor with resistance r.

The key aspect of this component-based modeling approach is that the component behavior is an *acausal* (see *relational*) description of the physical relationship among the effort and flow variables of the terminals. This is in contrast with the traditional *functional* modeling approach that requires to fix a-priori the input-output causality of an elementary building block on the base of a detailed global knowledge of the entire system. The fact that no functional dependencies are fixed a-priori in the components makes the acausal modeling approach particularly suitable to build a library of general components that are reusable in different designs. Figure 2.6 clarifies the difference between the acausal and the functional modeling approach on the resistor component. The acausal model on the left can be expanded in several equivalent functional models that differ for the input-output dependency between current and voltage (i.e. current computed from voltage or vice versa).

The main drawback of the functional modeling is that the elementary building block of the model are not reusable as with acausal modeling because the block causality strongly depends on the entire system design. Figure 2.7 presents an example of this main issue. The electrical networks on the left just differ for the voltage and current sources that drive the circuit. The acausal modeling approach allows the designer to reuse the same resistor component in both circuits, despite of the nature of the deployed power source, keeping a *one-to-one mapping* between the compo-

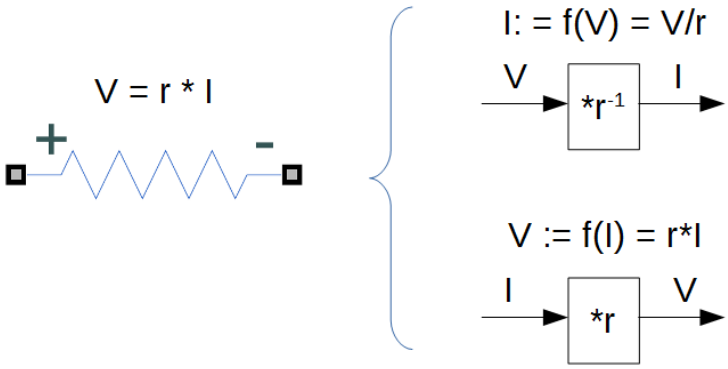


Figure 2.6: Ohm's law in its acausal form (on the left) and functional forms (on the right). The symbol V is a shortcut for the potential difference $v_+ - v_-$, while the symbol I is a shortcut for the current i_+.

nents of the real system and the components of the model. On the other side, the functional modeling approach requires to instantiate the resistor block with the correct causality depending on the particular source applied. In other words, the functional approach requires to explicitly unroll and hard-code the system causality into the system model making the design task exponentially more complex for large systems. On the other side, the interconnection of acausal components actually interconnects systems of equations in the individual components with one another. By interconnecting components, we do not define the calculation procedure (i.e. the global functional dependencies), but rather the modeled reality. The method of solving the equations is then "left to the machines" and we deal with this problem in this thesis.

2.2.2 Differential components

The electrical *capacitor* of Figure 2.8 introduces a differential behavior in the network of Figure 2.1. The capacitor is similar to a balloon that increases its internal pressure when is inflated, while decreases its pressure when is deflated. The *charge law* of the capacitor is $c * \frac{dV}{dt} = I$, where

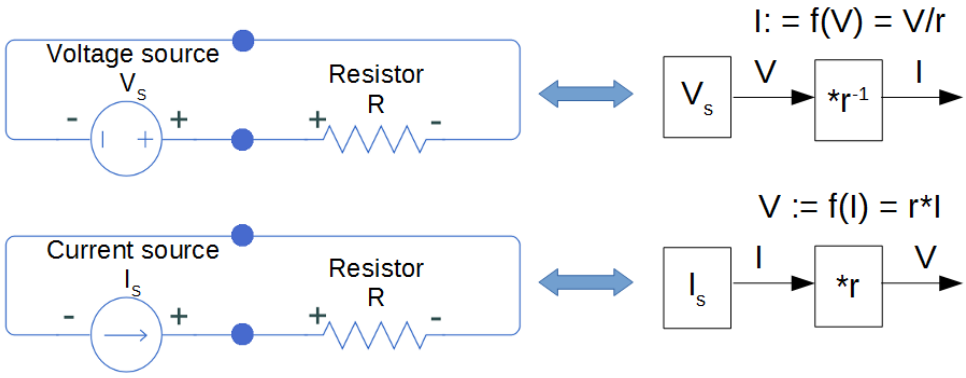


Figure 2.7: Comparison of the acausal (on the left) and functional (on the right) modeling approach for a simple power source-resistor network. The functional approach requires to unroll and hard-code into the model the functional dependencies between the variables. The causality of the resistor block depends on the nature of the power source.

Figure 2.8: Electrical capacitor.

V is a shortcut for the voltage drop $v_+ - v_-$ between the terminals, and I is a shortcut for the current i_+ through the positive terminal. This law says that the voltage drop V of the capacitor increases when the current I enters the positive terminal with a positive proportionality constant $1/c$, where the parameter c is called *capacitance*. Further details on the capacitor behavior are available in the Chapter A. Since the voltage drop V is implicitly defined by means of its first-derivative $\frac{dV}{dt}$, we need to solve some kind of differential equations starting from a known initial value $V(t_0)$ in order to know the time evolution of V. The fact that the present value of V depends on its past history makes the voltage drop V a *state* variable of the network. We highlight that in the capacitor component, the first-derivative $\frac{dV}{dt}$ is not directly provided as a function of the state variable V or of the input variables of the current source (i.e. it is not an ordinary-differential equations), but it is just related to the local current I that will be deter-

mined by the mutual-interaction of the capacitor with the interconnected network. In order to formally verify the network with the existing model checkers we need to make explicit the functional dependency between the *state* variable V, its first-derivative $\frac{dV}{dt}$ and the *input* variables of the network, performing what we call *reformulation* of the network dynamics into an equivalent ODE dynamics.

2.2.3 Switched components

Differently from the standard literature in mechatronic systems [Jan11], we focus on *Switched* Multi-Domain Kirchhoff Networks (SMDKN): differently from a non-switched MDKN, in a SMDKN each component further models a set of discrete states that can change instantaneously. A simple example of such components in the electrical domain is the *ideal switch* of Figure 2.9 that can be either in the *open* state or in the *closed* state, and changes its discrete state instantaneously (i.e., in a negligible physical time). In every discrete state, called *mode* in the following chapters, the component toggles between different physical behaviors, either enabling or disabling different sets of equations. In the closed mode,, the electrical switch is

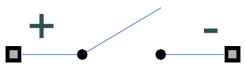


Figure 2.9: Electrical ideal switch.

equivalent to a short-circuit (i.e. zero voltage drop between terminals and current determined by the mutual-interaction with the network). In the open mode, the switch is equivalent to an open-circuit (i.e. zero current through the terminals and voltage drop between terminals determined by the mutual-interaction with the network). From the point of view of its equations, in the closed mode the switch enables the constraint $v_+ - v_- = 0$,

while in the open mode the switch enables the constraint $i_+ = 0$. Please refer to the Table 2.3 for a full list of the switch constraints and to the Chapter A for further details on the component.

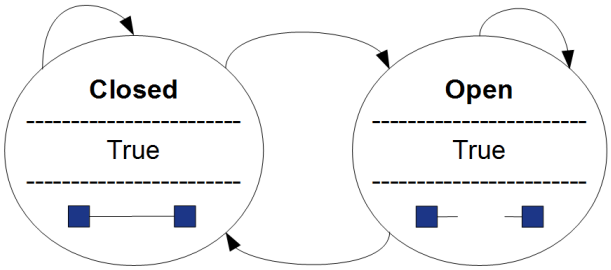


Figure 2.10: Hybrid automaton of the electrical ideal switch.

In terms of hybrid automata, the switch can be graphically represented as the two-location automaton of Figure 2.10. The automaton does not contain guards or switching rules on the discrete transitions because in the network of Figure 2.1 we consider switches that can toggle non-deterministically. Other switching components could define some switching rules to control when a transition must be fired depending on the physical state of the component. For instance, an electrical fuse is a kind of switch that starts in the closed mode, and permanently toggles to the open mode when the local current exceeds an overcurrent threshold.

2.3 DAE dynamics of the network

The introduction of discrete states complicates the analysis of the network, forcing the designer to reason about a number of network configurations that increases exponentially with the discrete modes of the components.

Consider the two distinct discrete configurations M_3 and M_7 of Figure 2.11 and Figure 2.12. They differ for the discrete mode of the switch S_0. In both configurations the switch S_1 is closed and the switch S_2 is open. In the configuration of Figure 2.11 the switch S_0 is open, while in

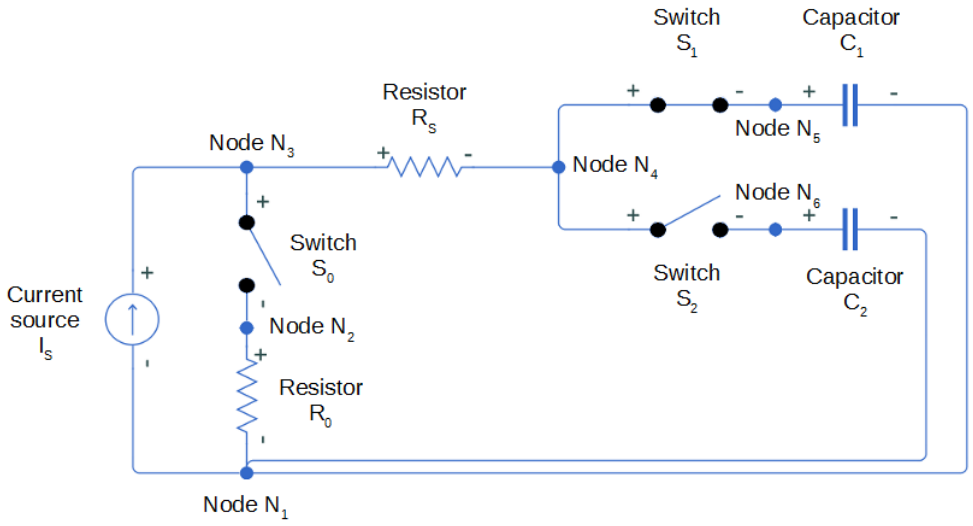


Figure 2.11: Network in the discrete configuration M_3 where switch S_0 is **open**, the switch S_1 is closed, and the switch S_2 is open. In M_3 the current source I_S charges the capacitor C_1 with a constant-rate current.

the configuration of Figure 2.12 the switch S_0 is closed. Fixing these global discrete configurations of the network, the physical behavior of the network is described by the sets of equations shown in Table 2.4. The two sets of constraints form two systems of differential-algebraic equations (DAE) that contains:

- all the non-switching constraints of Table 2.2 and Table 2.3;
- the switching constraints of Table 2.3 associated to the proper discrete modes of the switches S_0, S_1, and S_2.

The two DAEs just differ for one constraint of the switch S_0 that in the first case represents an open-circuit, while in the second case represents a short-circuit.

We cannot analyze this DAE continuous dynamics with the existing model checkers. Existing model checkers for hybrid automata accept continuous dynamics as either *explicit functions of time* or *implicit functions of time* in the form of differential equations. For formal analysis of SMDKN we

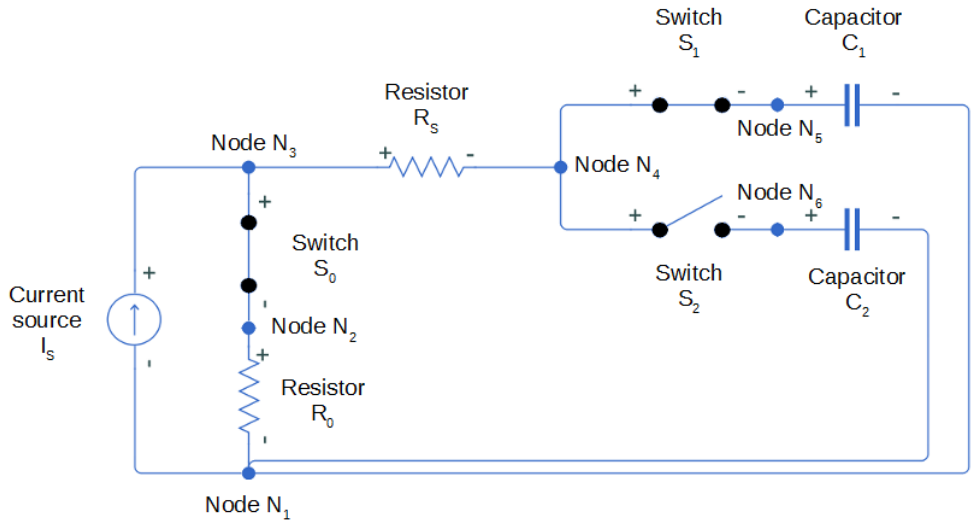


Figure 2.12: Network in the discrete configuration M_7 where switch S_0 is **closed**, the switch S_1 is closed, and the switch S_2 is open. In M_7 the current source I_S charges the capacitor C_1 with a constant-rate voltage.

are interested in differential dynamics. Model checkers allow the designer to specify different classes of differential dynamics from simple constant-rate dynamics to more complex ODE dynamics. Unfortunately, DAE dynamics are not supported. Since SMDKN contains DAE dynamics, our approach is based on a reformulation procedure of the DAE dynamics into ODE dynamics in order to model check SMDKN with the existing tools. Unfortunately, the DAE to ODE reformulation process is not always possible and this depends on the *structural properties* of the DAE. In Section 2.4, we describe the *reformulation process* of the network of Figure 2.1 for the discrete configurations of the network that admit a reformulation. Section 2.5 introduces the *validation problem* of the network that aims at discovering the discrete configurations that do not admit a reformulation.

	DAE of the discrete mode M_3	DAE of the discrete mode M_7
N_1	$I_S.i_- + R_0.i_- + C_1.i_- + C_2.i_- = 0$	$I_S.i_- + R_0.i_- + C_1.i_- + C_2.i_- = 0$
	$I_S.v_- = R_0.v_-$	$I_S.v_- = R_0.v_-$
	$I_S.v_- = C_1.v_-$	$I_S.v_- = C_1.v_-$
	$I_S.v_- = C_2.v_-$	$I_S.v_- = C_2.v_-$
N_2	$R_0.i_+ + S_0.i_- = 0$	$R_0.i_+ + S_0.i_- = 0$
	$R_0.v_+ = S_0.v_-$	$R_0.v_+ = S_0.v_-$
N_3	$I_S.i_+ + S_0.i_+ + R_S.i_+ = 0$	$I_S.i_+ + S_0.i_+ + R_S.i_+ = 0$
	$I_S.v_+ = S_0.v_+$	$I_S.v_+ = S_0.v_+$
	$I_S.v_+ = R_S.v_+$	$I_S.v_+ = R_S.v_+$
N_4	$R_S.i_- + S_1.i_+ + S_2.i_+ = 0$	$R_S.i_- + S_1.i_+ + S_2.i_+ = 0$
	$R_S.v_- = S_1.v_+$	$R_S.v_- = S_1.v_+$
	$R_S.v_- = S_2.v_+$	$R_S.v_- = S_2.v_+$
N_5	$S_1.i_- + C_1.i_+ = 0$	$S_1.i_- + C_1.i_+ = 0$
	$S_1.v_- = C_1.v_+$	$S_1.v_- = C_1.v_+$
N_6	$S_2.i_- + C_2.i_+ = 0$	$S_2.i_- + C_2.i_+ = 0$
	$S_2.v_- = C_2.v_+$	$S_2.v_- = C_2.v_+$
I_S	$I_S.i_- + I_S.i_+ = 0$	$I_S.i_- + I_S.i_+ = 0$
	$I_S.i_- = i_s$	$I_S.i_- = i_s$
R_0	$R_0.i_- + R_0.i_+ = 0$	$R_0.i_- + R_0.i_+ = 0$
	$R_0.v_+ - R_0.v_- = V_{R_0}$	$R_0.v_+ - R_0.v_- = V_{R_0}$
	$V_{R_0} = r_0 * R_0.i_+$	$V_{R_0} = r_0 * R_0.i_+$
R_S	$R_S.i_- + R_S.i_+ = 0$	$R_S.i_- + R_S.i_+ = 0$
	$R_S.v_+ - R_S.v_- = V_{R_S}$	$R_S.v_+ - R_S.v_- = V_{R_S}$
	$V_{R_S} = r_S * R_S.i_+$	$V_{R_S} = r_S * R_S.i_+$
S_0	$S_0.i_- + S_0.i_+ = 0$	$S_0.i_- + S_0.i_+ = 0$
	$S_0.i_+ = 0$	$S_0.v_+ - S_0.v_- = 0$
S_1	$S_1.i_- + S_1.i_+ = 0$	$S_1.i_- + S_1.i_+ = 0$
	$S_1.v_+ - S_1.v_- = 0$	$S_1.v_+ - S_1.v_- = 0$
S_2	$S_2.i_- + S_2.i_+ = 0$	$S_2.i_- + S_2.i_+ = 0$
	$S_2.i_+ = 0$	$S_2.i_+ = 0$
C_1	$C_1.i_- + C_1.i_+ = 0$	$C_1.i_- + C_1.i_+ = 0$
	$C_1.v_+ - C_1.v_- = V_{C_1}$	$C_1.v_+ - C_1.v_- = V_{C_1}$
	$c_1 * \frac{dV_{C_1}}{dt} = C_1.i_+$	$c_1 * \frac{dV_{C_1}}{dt} = C_1.i_+$
C_2	$C_2.i_- + C_2.i_+ = 0$	$C_2.i_- + C_2.i_+ = 0$
	$C_2.v_+ - C_2.v_- = V_{C_2}$	$C_2.v_+ - C_2.v_- = V_{C_2}$
	$c_2 * \frac{dV_{C_2}}{dt} = C_2.i_+$	$c_2 * \frac{dV_{C_2}}{dt} = C_2.i_+$

Table 2.4: DAE dynamics of the network of Figure 2.1 in two different discrete configurations: S_0 **open**, S_1 closed, S_2 open (discrete mode M_3), and S_0 **closed**, S_1 closed, S_2 open (discrete mode M_7). The DAE dynamics just differ for the highlighted constraint of the switch S_0.

2.4 Network reformulation

Given the DAE dynamics of a discrete configuration of the network that admits a reformulation, the problem of computing the reformulation consists in finding a *unique functional rewriting* (i.e. the ODE) of the first-derivative variables in terms of the *input* and *state* variables of the network. This functional rewriting must also be *coherent* with the original DAE for every value of the input and state variables. When the ODE function over the first-derivative, state and input variables is found, the remaining algebraic variables of the DAE can be expressed as an *algebraic relation* between the input, state and algebraic variables by means of a *syntactic replacement* of the first-derivatives in the DAE with their functional reformulations over state and input variables. The reformulation approach based on the algebraic relation allows us to model and reason on possible non-deterministic behavior of the algebraic variables.

For the network of Figure 2.1 only five of the eight possible discrete configurations admit an ODE reformulation. Table 2.5 shows the ODE reformulations for the five configuration. We recall from Table 2.4 that the DAE of the discrete modes M_3 and M_7 just differ for one constraint. Although this is a small difference, the reformulations of modes M_3 and M_7 fall into two distinct classes of continuous dynamics because M_3 has a *constant-rate* dynamics while M_7 has an ODE dynamics. It is important to take into consideration the class of the reformulation because different existing model checkers provide verification algorithms specialized for particular continuous dynamics.

Mode	Switch configurations	Reformulation	Type
M_2	S_0 open, S_1 open, S_2 closed	$\frac{dV_{C_1}}{dt} = 0.0$ $\frac{dV_{C_2}}{dt} = 0.5 * I_S.i\_$	Constant-rate
M_3	S_0 open, S_1 closed, S_2 open	$\frac{dV_{C_1}}{dt} = 0.5 * I_S.i\_$ $\frac{dV_{C_2}}{dt} = 0.0$	Constant-rate
M_5	S_0 closed, S_1 open, S_2 open	$\frac{dV_{C_1}}{dt} = 0.0$ $\frac{dV_{C_2}}{dt} = 0.0$	Constant-rate
M_6	S_0 closed, S_1 open, S_2 closed	$\frac{dV_{C_1}}{dt} = 0.0$ $\frac{dV_{C_2}}{dt} = -0.25 * V_{C_2} + 0.25 * I_S.i\_$	ODE
M_7	S_0 closed, S_1 closed, S_2 open	$\frac{dV_{C_1}}{dt} = -0.25 * V_{C_1} + 0.25 * I_S.i\_$ $\frac{dV_{C_2}}{dt} = 0.0$	ODE

Table 2.5: ODE reformulations of the network of Figure 2.1 for the five discrete configurations that admit the reformulation. The ODE reformulations refers to the following parameters assignments: $r_0 = r_S = 1.0\Omega$, $c_1 = c_2 = 2.0F$.

Given the ODE reformulation of the first-derivative variables $\frac{dV_{C_1}}{dt}$ and $\frac{dV_{C_2}}{dt}$, we compute the *algebraic relation* of a discrete mode replacing the first-derivatives with their functional representation into the DAE equations of the network. Table 2.6 shows an example of this syntactic replacement for the mode M_7. SMT engines are very efficient in reasoning on this kind of relations and we use this approach to avoid to compute the functional reformulation of all the algebraic variables. Moreover, the algebraic relation allows us to model non-deterministic algebraic variables that would be impossible to express in a functional way.

	DAE of the discrete mode M_7	algebraic relation of the discrete mode M_7
N_1	$I_S.i_- + R_0.i_- + C_1.i_- + C_2.i_- = 0$	$I_S.i_- + R_0.i_- + C_1.i_- + C_2.i_- = 0$
	$I_S.v_- = R_0.v_-$	$I_S.v_- = R_0.v_-$
	$I_S.v_- = C_1.v_-$	$I_S.v_- = C_1.v_-$
	$I_S.v_- = C_2.v_-$	$I_S.v_- = C_2.v_-$
N_2	$R_0.i_+ + S_0.i_- = 0$	$R_0.i_+ + S_0.i_- = 0$
	$R_0.v_+ = S_0.v_-$	$R_0.v_+ = S_0.v_-$
N_3	$I_S.i_+ + S_0.i_+ + R_S.i_+ = 0$	$I_S.i_+ + S_0.i_+ + R_S.i_+ = 0$
	$I_S.v_+ = S_0.v_+$	$I_S.v_+ = S_0.v_+$
	$I_S.v_+ = R_S.v_+$	$I_S.v_+ = R_S.v_+$
N_4	$R_S.i_- + S_1.i_+ + S_2.i_+ = 0$	$R_S.i_- + S_1.i_+ + S_2.i_+ = 0$
	$R_S.v_- = S_1.v_+$	$R_S.v_- = S_1.v_+$
	$R_S.v_- = S_2.v_+$	$R_S.v_- = S_2.v_+$
N_5	$S_1.i_- + C_1.i_+ = 0$	$S_1.i_- + C_1.i_+ = 0$
	$S_1.v_- = C_1.v_+$	$S_1.v_- = C_1.v_+$
N_6	$S_2.i_- + C_2.i_+ = 0$	$S_2.i_- + C_2.i_+ = 0$
	$S_2.v_- = C_2.v_+$	$S_2.v_- = C_2.v_+$
I_S	$I_S.i_- + I_S.i_+ = 0$	$I_S.i_- + I_S.i_+ = 0$
	$I_S.i_- = i_s$	$I_S.i_- = i_s$
R_0	$R_0.i_- + R_0.i_+ = 0$	$R_0.i_- + R_0.i_+ = 0$
	$R_0.v_+ - R_0.v_- = V_{R_0}$	$R_0.v_+ - R_0.v_- = V_{R_0}$
	$V_{R_0} = r_0 * R_0.i_+$	$V_{R_0} = r_0 * R_0.i_+$
R_S	$R_S.i_- + R_S.i_+ = 0$	$R_S.i_- + R_S.i_+ = 0$
	$R_S.v_+ - R_S.v_- = V_{R_S}$	$R_S.v_+ - R_S.v_- = V_{R_S}$
	$V_{R_S} = r_S * R_S.i_+$	$V_{R_S} = r_S * R_S.i_+$
S_0	$S_0.i_- + S_0.i_+ = 0$	$S_0.i_- + S_0.i_+ = 0$
	$S_0.v_+ - S_0.v_- = 0$	$S_0.v_+ - S_0.v_- = 0$
S_1	$S_1.i_- + S_1.i_+ = 0$	$S_1.i_- + S_1.i_+ = 0$
	$S_1.v_+ - S_1.v_- = 0$	$S_1.v_+ - S_1.v_- = 0$
S_2	$S_2.i_- + S_2.i_+ = 0$	$S_2.i_- + S_2.i_+ = 0$
	$S_2.i_+ = 0$	$S_2.i_+ = 0$
C_1	$C_1.i_- + C_1.i_+ = 0$	$C_1.i_- + C_1.i_+ = 0$
	$C_1.v_+ - C_1.v_- = V_{C_1}$	$C_1.v_+ - C_1.v_- = V_{C_1}$
	$c_1 * \frac{dV_{C_1}}{dt} = C_1.i_+$	$c_1 * (-0.25 * V_{C_1} + 0.25 * I_S.i_-) = C_1.i_+$
C_2	$C_2.i_- + C_2.i_+ = 0$	$C_2.i_- + C_2.i_+ = 0$
	$C_2.v_+ - C_2.v_- = V_{C_2}$	$C_2.v_+ - C_2.v_- = V_{C_2}$
	$c_2 * \frac{dV_{C_2}}{dt} = C_2.i_+$	$c_2 * 0.0 = C_2.i_+$

Table 2.6: Computation of the algebraic relation of the discrete configuration M_7. The first-derivative variables ($\frac{dV_{C_1}}{dt}$, $\frac{dV_{C_2}}{dt}$) in the DAE are syntactically replaced with their ODE reformulation in terms of state variables (V_{C_1}, V_{C_2}) and input variables ($I_S.i_-$).

2.4.1 Shared reformulations

Let us focus on the rows of the reformulations of Table 2.5, namely on the reformulation of a specific first-derivative variable (e.g. $\frac{dV_{C_1}}{dt}$). Although there are five discrete modes that admit a reformulation, there are only three distinct reformulations of $\frac{dV_{C_1}}{dt}$. In fact, the three modes M_2, M_5 and M_6 share the same reformulation $\frac{dV_{C_1}}{dt} = 0.0$. This is due to the structural symmetries of the network that produce a clustering of the theoretical continuous dynamics in only few distinct dynamics. In particular, the three modes M_2, M_5 and M_6 share the same dynamics for $\frac{dV_{C_1}}{dt}$ because they share the same open position of the switch S_1. When S_1 is open, despite of the position of the other switches, the capacitor C_1 is neither charged nor discharged, thus M_2, M_5 and M_6 collapse in a macro-configuration with the same continuous dynamics. Similarly, the modes M_3, M_5 and M_7 share the same reformulation $\frac{dV_{C_2}}{dt} = 0.0$ for $\frac{dV_{C_2}}{dt}$.

Since SMT-based techniques are able to efficiently reason on this kind of structural symmetries, in our approach we perform the reformulation of the DAE dynamics by ODE-rows in order to exploit the network symmetries and speed-up the reformulation process.

2.5 Network validation

2.5.1 Inconsistent configurations

The network of Figure 2.1 does not admit an ODE reformulation in the discrete configurations M_1, M_4 and M_8. Figure 2.13 shows the discrete configuration M_4. When both the switches S_1 and S_2 are closed, the two capacitors C_1 and C_2 are connected in a loop, and, according to standard results from the electrical theory [Ria08], this loop prevents the existence

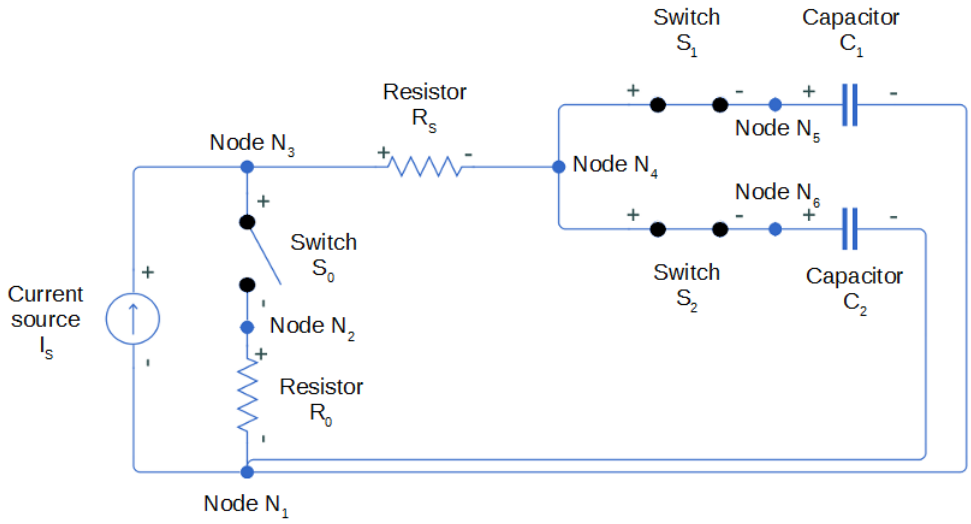


Figure 2.13: Network in the discrete configuration M_4 where the switch S_0 is open, the switch S_1 is **closed**, and the switch S_2 is **closed**. In M_4 the capacitors C_1 and C_2 form a capacitor-loop (generally called VC-loop) that prevents the existence of an ODE reformulation.

of the ODE reformulation for the mode M_4 because the two state variables V_{C_1} and V_{C_2} become mutually-dependent. From the physical point of view, the parallel connection of two capacitors produces a fast electrical transient with over-currents that could damage the system. The capacitor-loop equivalently affects the discrete configuration M_8.

The discrete configuration M_1 (shown in Figure 2.1) exhibits a inconsistency of different nature that involves the current source and its input variable $I_S.i\_$. When all the switches are open, the current source I_S is "obstructed", thus the DAE equations are inconsistent.

For all these kinds of inconsistencies, the structure of the network in all its discrete configurations needs to be analyzed before to perform the reformulation. Our validation technique aims to spot and report to the designer the harmful network configurations in the earlier stages of the design process to prevent system failures in operation, and to guarantee the correctness of the reformulation. The validation check is unfeasible

in practice due to the exponential blow-up of discrete configurations and to the alternation of universal and existential quantifiers involved in the check. Our validation approach exploits the algebraic structure of the problem to circumvent the nested quantifier elimination and to brake the problem into simpler checks that rely on a single existential quantifier. In the following section we explain how the check applies to the general case of Switched Kirchhoff Network and why it is equivalent to the nested quantifier elimination.

2.5.2 Under-specified configurations

A key aspect of the acausal modeling approach is that the physics of an elementary components is by construction *under-specified*. In fact, the equations of the component do not determine the actual behavior of all its variables. They are (possibly) determined by the mutual-interaction of the component with the interconnected network. Extending this concept to the network, when a network is *under-specified* it does not contain a sufficient amount of physical constraints w.r.t. its variables and some of them might remain only partially determined. The designer must deal with this aspect of the modeling to ensure the correctness of the analysis.

All the discrete configurations of the network of Figure 2.1 are under-specified. In fact, the lack of a reference potential in the network (i.e. an electrical ground component) make impossible to determine an exact value of the terminal potentials w.r.t. an absolute value (e.g. the ground reference value). Nevertheless, the network is fully-specified in terms of branch currents and voltage drops between terminals. Since the ODE reformulation is expressed in terms of the fully-specified variables, we can perform the reformulation of the five consistent modes M_2, M_3, M_5, M_6, and M_7, keeping the under-specified variables in the algebraic relation that can handle their non-determinism. The network of Figure 2.14 adds the miss-

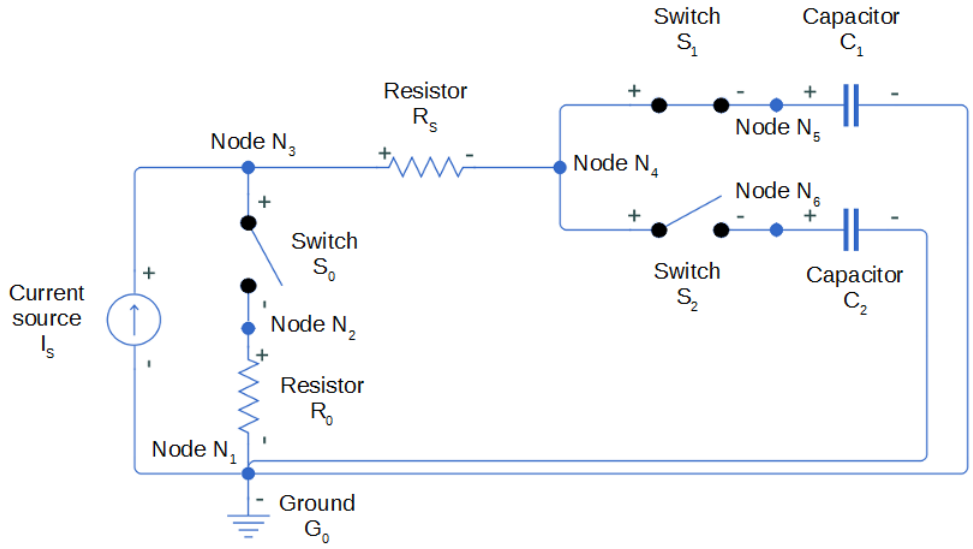


Figure 2.14: Network in the discrete configuration M_3 with the addition of the Ground component G_0 that makes **deterministic** the potential of all the terminals.

ing ground component G_0 that makes the network fully-determined. In a general SMDKN there are several sources of non-determinism that can possibly propagate to the ODE reformulation variables. In order to guarantee the correctness of the ODE reformulation, we propose a validation procedure that discovers and reports to the designer all the under-specification of the network in order to fix the network models if necessary. In general, the under-specification test checks whether a variable admits *at most one* solution *for every* value of the input and state variables. This check requires nested quantifier elimination. Our validation approach exploits the algebraic structure of the problem to circumvent the nested quantifier elimination and to reduce the validation problem to a single existential quantification.

Chapter 3

Background

In the following we provide the necessary background on Satisfiability Modulo Theory, Differential Algebraic Equations, and Hybrid Automata.

3.1 General Notation

We use $\mathbb{R}$ to denote the set of Real numbers. We denote the cardinality of a set X with $|X|$ and its powerset (i.e., the set of all X's subsets) as 2^X.

We denote with X' the set obtained from the set X replacing each variable x with its "primed" version x' (i.e., $X' := \{x' | x \in X\}$) and, analogously, we denote with $\dot{X}$ the set obtained from X replacing each variable x with its first derivative $\dot{x}$ (i.e., $\dot{X} := \{\dot{x} | x \in X\}$).

We further use the notation $\vec{x}$ to refer to the column-vector containing all the variables in X ordered lexicographically (e.g., for the set $X := \{x_0, x_1\}$, its vector representation is $\vec{x} := \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}$). We will use both notations depending on the context since X is more convenient when working with first-order logic formulas, while $\vec{x}$ is more convenient when working with the matrix equation form of a system of linear equations.

3.2 Differential Algebraic Equations

We model the continuous evolution of a dynamical systems with respect to time as differential equations. We first define the class of Ordinary Differential Equations (ODEs) and then the more general Differential Algebraic Equations (DAEs). We further recall the problem of expressing a system of Differential Algebraic Equations as a system of Ordinary Differential Equations, restricting our focus to linear ODEs and DAEs.

3.2.1 Ordinary Differential Equations

A system of (linear) *Ordinary Differential Equations (ODEs)* defines the first-derivative of a vector or real valued functions $\vec{x} : (\mathbb{R} \to \mathbb{R})^n$:

$$\dot{\vec{x}}(t) = \vec{A}\,\vec{x}(t) + \vec{B}\,\vec{u}(t)$$

where $\dot{\vec{x}}(t)$ is the column vector of first-derivatives with respect to time of the *state* functions $\vec{x}(t)$, $\vec{u} \in (\mathbb{R} \to \mathbb{R})^m$ is a vector of *input* functions, $\vec{A} \in \mathbb{R}^{|\vec{x}|\times|\vec{x}|}$ and $\vec{B} \in \mathbb{R}^{|\vec{x}|\times|\vec{u}|}$ are constant matrices, and t represents the independent variable (time, in our case). In the following, we drop the explicit dependence from time from $\vec{x}$ and $\vec{u}$ and we refer to them as state and input variables, respectively. A solution of the system of ODEs is a vector of functions $\vec{x}(t)$ that satisfies the system of equations. The system of ODEs $\dot{\vec{x}}(t) = \vec{A}\,\vec{x}(t) + \vec{B}\,\vec{u}(t)$ with continuous input functions $u_i(t)$ has a unique solution to the initial value problem with $\vec{x}(t_0) = \vec{x_0}$ for all possible times $t_0 > 0$.

3.2.2 Differential-Algebraic Equations

A system of *Differential-Algebraic Equations* DAEs is a system of equations containing state variables $\vec{x}$, their derivatives $\dot{\vec{x}}$, input variables $\vec{u}$, and algebraic variables $\vec{y}$. The *algebraic variables* $\vec{y}$ evolve continuously in

time, as the state variables, but their derivative is expressed implicitly in the other constraints of the DAEs (i.e., their derivative never appears in the equations). We define a linear system of DAEs as follows:

$$\vec{M}\,\dot{\vec{x}} + \vec{N}\,\vec{x} + \vec{O}\,\vec{y} + \vec{P}\,\vec{u} = \vec{0} \tag{3.1}$$

where $\vec{x}$, $\vec{y}$, and $\vec{u}$ are the state, algebraic, and input variables, and $\vec{M}, \vec{N}, \vec{O}, \vec{P}$ are constants real matrices.

Reasoning about DAEs (e.g., simulating the DAEs given an initial value, computing the set of reachable states from an initial set of states) is further challenging due to the algebraic constraints, which in turn impose implicit constraints on the differential equations of the variables. In practice, for both simulation and verification we aim to finding a representation of the DAEs system of the form:

$$\begin{aligned}\dot{\vec{x}} =& \vec{A}\,\vec{x} + \vec{B}\,\vec{u}\\ \vec{y} =& \vec{C}\,\vec{x} + \vec{D}\,\vec{u}\end{aligned}$$

In several cases we can obtain the above representation automatically from the DAE formulation of Equation 3.1, through a process called *structural analysis*. For example, when the matrix $\vec{M}$ from Equation 3.1 is non-singular we can obtain a system of ODEs for the state variables $\vec{x}$ just inverting the matrix $\vec{M}$. In this case, we say that the system of DAEs is of structural index-1. If the index is greater than 1 (e.g., when the matrix $\vec{M}$ is non-singular) we have to apply an index-reduction algorithm (e.g., [Pry01]) trying to reduce the DAEs index using symbolic differentiation, eventually getting an index-1 DAEs (and hence an ODEs). In this thesis, we focus on systems of DAEs of index-1.

Additionally, we relax the requirement for the functional form $\vec{y} = \vec{C}\,\vec{x} + \vec{D}\,\vec{u}$ of the algebraic variables $\vec{y}$ allowing the algebraic variables to be expressed as a more general relation $\phi(\vec{y}, \vec{x}, \vec{u})$ that admits non-deterministic behaviors.

3.3 Satisfiability Modulo Theories

Our setting is standard first-order logic. Let Σ be a first-order *signature* containing predicates and function symbols with their arity and X be a set of variables. A 0-ary predicate symbol A is called a *Boolean atom* while a 0-ary function symbol c is called a *constant*. A Σ-*term* is either a variable or it is built applying function symbols in Σ to Σ-terms. If p is a predicate with arity n and $t_1, \ldots, t_n$ are Σ-terms, then $p(t_1, \ldots, t_n)$ is a Σ-*atom*. A Σ-*formula* is either a Σ-atom, or the application of the Boolean connectives $\wedge, \vee, \neg$ to two Σ-formulas, or the application of quantifiers ($\exists$, $\forall$) to an individual variable and a Σ-formula.

We use the standard abbreviations for the other Boolean operators, $\phi_1 \to \phi_2$ in place of $\neg\phi_1 \vee \phi_2$, and $\phi_1 \leftrightarrow \phi_2$ in place of $(\phi_1 \to \phi_2) \wedge (\phi_2 \to \phi_1)$. We further use the abbreviations $\exists X.\phi$ and $\exists \vec{x}.\phi$, where ϕ is a Σ-formula, $X := \{x_1, \ldots, x_n\}$ is a set of variables, and $\vec{x} := [x_0, \ldots, x_n]^T$ is a vector of variables, in place of the formula $\exists x_1. \ \ldots \ \exists x_n.\phi$.

A variable is *free* in a formula ϕ if it is not quantified, and is *bound* otherwise. Given a formula ϕ we write $\phi(X)$ to denote that X is the set of free variables in ϕ. A *sentence* with signature Σ is a Σ-formula without free variables. A first-order Σ-*theory* $\mathcal{T}$ is a set of first-order sentences with signature Σ. We assume that the symbols identity $=$, false $\bot$, and true $\top$ are always part of the language.

We assume the standard first-order notion of interpretation, satisfiability, validity, and logical consequence. We write $\phi[s/t]$ for the formula obtained from the formula ϕ substituting all the occurrences of the term t with the term s. We use the abbreviation $\phi[\vec{s}/\vec{t}]$ where $\vec{s}$ and $\vec{t}$ have the same length to denote the element-wise substitution of every occurrence of the i-th element of $\vec{t}$ with the i-th element of $\vec{s}$. We call μ a *satisfying assignment* or model of a formula $\phi(X)$ a total function that assigns

to every $x \in X$ an element c in the x's domain such that the formula $\phi[\mu(X)/X]$ evaluates to $\top$ provided an interpretation of the function symbols. A formula ϕ is $\mathcal{T}$*-satisfiable* if there exists an interpretation and an assignment μ such that $\phi[\mu(X)/X]$ evaluates to $\top$. Two formulas ϕ_1 and ϕ_2 are $\mathcal{T}$*-equisatisfiable* if and only if ϕ_1 is $\mathcal{T}$-satisfiable if and only if ϕ_2 is $\mathcal{T}$-satisfiable.

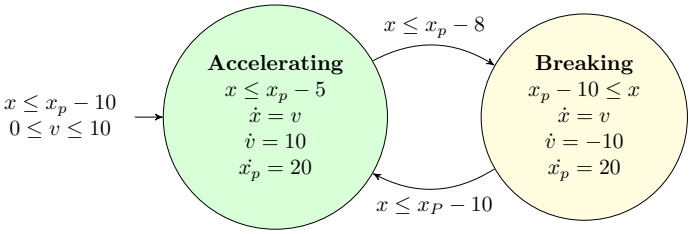
The *Satisfiability Modulo Theory* problem (SMT ($\mathcal{T}$)) [BSST09] is the problem of deciding if a formula ϕ is $\mathcal{T}$-satisfiable.

SMT can be seen as an extension of Boolean satisfiability (SAT), where literals are interpreted with respect to a background theory $\mathcal{T}$. In this work, we interpret the formulas in the *Linear Arithmetic* on the rationals (LRA). Its signature is $\Sigma_{\mathbb{R}} = \{0, 1, +, -, =, \geq\}$, where 0 and 1 are the rational number constants, $+, -$ are the usual addition and subtraction operators, and $=, \geq$ are the equal and greater than or equal relational operators. The resulting language consists of the formulas with atoms in the form $\sum_i a_i x_i \bowtie a$, where x_i is a variable, $a_i, a \in \mathbb{R}$ are real constants, and $\bowtie \in \{<, \leq, >, \geq, =, \neq\}$ is a relational operator. Note that the satisfiability of the LRA theory is decidable. In the following, we drop the LRA suffix and write $\Gamma \models \phi$ instead of $\Gamma \models_{LRA} \phi$. We further write $\mu_{|X}$ for the function obtained restricting the domain of the assignment μ to the variables X.

3.4 Hybrid automata

Hybrid automata (HA) [Hen96] are a mathematical formalism for modeling dynamical systems in which instantaneous computations interact with continuous physical processes. A hybrid automaton is a state machine enriched with a finite set of continuous variables whose time evolution is described by a set of differential equations. In Figure 3.1a we show the hybrid automaton for a simplified cruise control system of a vehicle that

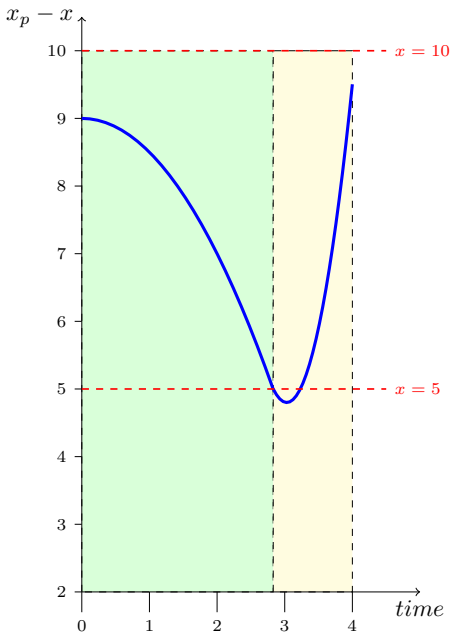
either accelerates or brakes. The model represents the position and velocity of the vehicle (variables x and v), and the position of the preceding vehicle (x_p). The automaton's modes model these two operating modes (*Accelerating* and *Breaking*), defining a different dynamic for the vehicle in each one of them. For example, in the *Accelerating* mode the system's dynamic is specified with the system of ODEs: $\dot{x} = v$, $\dot{v} = 10$, and $\dot{x_p} = 20$. Furthermore, the invariant conditions (e.g., $x \leq x_p - 5$ in the *Accelerating* mode) define when the system can stay in a mode or not. Figure 3.1c shows the difference between x_p and x in a possible trajectory of the automaton. The trajectory shows that the automaton switches from the *Accelerating* to the *Braking* mode (when $x = 5$) forcing the vehicle to break.



(a) Hybrid Automaton of the cruise control system.

$$\begin{aligned}
&\langle \{m\}, \{x, v, x_p\}, x \leq x_p - 10 \wedge 0 \leq v \leq 10, \\
&\quad (m \rightarrow x \leq x_p - 5) \wedge (\neg m \rightarrow x - 10 \leq x_p), \\
&\quad (m \wedge \neg m' \wedge x \leq x_p - 8) \vee \\
&\qquad (\neg m \wedge m' \wedge x \leq x_p - 10), \\
&\quad \dot{x_p} = 20 \wedge \dot{x} = v \wedge \\
&\qquad (m \rightarrow \dot{v} = 10) \wedge (\neg m \rightarrow \dot{v} = -10) \rangle
\end{aligned}$$

(b) Symbolic representation.



(c) Difference $x_p - x$ for a trajectory of the hybrid automaton.

Figure 3.1: Explicit hybrid automaton, symbolic hybrid automaton, and trajectory of a cruise control system of a vehicle. The variables x, v, and x_p represent the position of the vehicle, its velocity, and the position of the preceding vehicle. The preceding vehicle moves with a constant velocity ($20m/s$), while the follower vehicle either accelerates (in the *Accelerating* mode), or brakes (in the *Braking* mode).

In the following, we use a symbolic representation of hybrid automata [CMT14]. We represent the states of the automaton with Boolean and Real valued variables, and set of states as Σ-formulas $\phi(X)$ on such variables. An assignment μ to the formula $\phi(X)$ then represents a *state* of the hybrid automaton. Analogoulsy, we represent the transition relation in the automaton as a formula $\phi(X, X')$. In this case, an assignment μ to the formula $\phi(X, X')$ represents a possible transition in the automaton. We further specify differential equations using Σ-formulas — however, their interpretation is given in terms of "runs" of the hybrid automaton and not in terms of the LRA theory[1]. A hybrid automaton H is the tuple:

$$H := \langle B, R, Init, Invar, Trans, Flow \rangle$$

where:

- B is a set of Boolean variables encoding the discrete modes of the automaton.
- R is a set of Real variables encoding the continuous variables of the automaton.
- $Init(B, R)$ is a $\Sigma_{\mathbb{R}}$-formula that represents the set of initial states.
- $Invar(B, R)$ is a $\Sigma_{\mathbb{R}}$-formula that represents the set of invariant states.
- $Trans(B, R, B', R')$ is a $\Sigma_{\mathbb{R}}$-formula that represents the set of discrete transitions.
- $Flow(B, \dot{R}, R)$ is a $\Sigma_{\mathbb{R}}$-formula that represents the differential constraints on the continuous variables.

[1]While it is possible to formulate a fisrt-order theory of Ordinary Differential Equations, for example following [GKC13b], here we take a different approach where the differential equations are taken into account when defining the hybrid automata semantic, and not when defining the interpretation of the first-logic formula.

Figure 3.1b shows the symbolic representation of the (non symbolic) hybrid automaton shown in Figure 3.1a. We see that the symbolic representation may be more concise than the non-symbolic representation, for example when specifying that $\dot{x} = v$ for all the possible modes.

While in this thesis we assume that all the formulas $Init$, $Invar$, $Trans$ and $Flow$ are quantifier-free formulas in the LRA Theory, the framework is more general and can take into account other theories (e.g., Uninterpreted Functions (UFs)).

A *state* of the hybrid automaton H is an assignment to the variables $B \cup R$. For example, the state $s_0 = \{x \mapsto 1, v \mapsto 20, x_p \mapsto 10\}$ is the initial state for the trajectory shown in Figure 3.1c.

Definition 1 (Hybrid Automaton Run.)**.** *A* run π *of the hybrid automaton* H *is a sequence of states* $\pi := s_0 \xrightarrow{\delta_1} s_1 \xrightarrow{\delta_2} \ldots \xrightarrow{\delta_k} s_k$ *such that all the following conditions hold:*

- $s_0 \models Init$ *and for all* $0 < i \leq k$, s_i *is a state of* H*.*
- *For* $1 \leq i \leq k$, $\delta_j \in \mathbb{R}$ *and* $s_{i-1} \xrightarrow{\delta_i} s_i$ *we have either a:*
  - *discrete transition:* $\delta_i = 0$ *and* $\langle s_{i-1}, \delta_i, s_i \rangle \models Trans$, $s_{i-i} \models Invar$*, and* $s_i \models Invar$*.*
  - *continuous transition:* $\delta_i > 0$ *and:*
    - $s_{i-1_{|V}} = s_{i_{|V}}$,
    - *there exists a continuous differentiable function* $f : [0, \delta_i] \rightarrow \mathbb{R}^{|R|}$ *such that:*
      - $f(0) = s_{i-1_{|R}}$ *and* $f(\delta_i) = s_{i_{|R}}$,
      - $s_{i-1} \models Invar$, $s_i \models Invar$,
      - $\forall \epsilon \in [0, \delta_i]$, $\langle s_{i-1_{|B}}, f(\epsilon), \dot{f}(\epsilon) \rangle \models Flow$,
      - $\forall \epsilon \in [0, \delta_i], \langle s_{i-1_{|B}}, f(\epsilon) \rangle \models Invar$.

A *run* is a sequence of states such that the first state is in the set of states specified in the initial condition $Init$, every state belongs to the states specified in the invariant condition $Invar$, and each pair of consecutive states either satisfies the transition relation in a discrete transition specified in $Trans$, or the differential constraints specified in $Flow$. Also, the continuous transition ensures that the invariant condition holds at every possible instant in time.

The semantics of the HA H is defined by the set of all its runs $[\![H]\!]$. Two hybrid automata H_1 and H_2 are *equivalent* if they accept the same runs.

We say that the hybrid automaton has an *ODE dynamics* if, for each discrete mode the flow condition takes the form of a system of ODEs. Otherwise, the hybrid automaton has a *DAE dynamics* if the flow condition takes the form of a systems of DAEs.

We are interested in the *Safety Verification Problem* for hybrid automata. A hybrid automaton H reaches a state s if there is a run $\pi := s_0 \xrightarrow{\delta_1} s_1 \xrightarrow{\delta_2} \ldots \xrightarrow{\delta_k} s_k$ such that $s_k = s$. The hybrid automaton H satisfies the safety property P if there are no runs $\pi \in [\![H]\!]$ such that π reaches a state $s \notin P$. That is, there are no runs of H that can reach an "unsafe" state, a state outside the safety property P.

Chapter 4

Formalizing Switched Multi-Domain Kirchhoff Networks

In the following, we describe the syntax and semantics of Smdkn.

4.1 Defining the Smdkn Syntax

To define the syntax of Switched Multi-Domain Kirchhoff Networks we first define the syntax of its components, and then we define how we connect the terminals of the components to form a network.

The definition of a network component describes its internal state, both the "physical" continuous state and the "digital" discrete state, and their dynamics, how the component's state changes either when time elapses or when the network takes an instantaneous action. The component's definition further describes its terminals, its interface with the network. While our component's definition is similar to the syntactic definition of a Hybrid Automaton, the two definitions differ in that a component carries more structure (e.g., the algebraic variables, the terminals, ...) and that we do not provide an execution semantic for a single component, but rather the semantic for the whole network. The definition of the component's semantic is difficult to provide due to the "dangling" terminals and the

algebraic differential equations.

Definition 2 (Network Component)**.** *A* network component *c is a tuple*

$$c := \langle B, R, T_{\mathcal{C}}, Init, Invar, Trans, Flow, Input \rangle$$

where:

- *B is a set of* Boolean variables *describing the discrete state.*
- *$R := X \cup U \cup Y$ is the set of* Continuous variables *that represents the physical quantities of the component and that can change value when time elapses.*

  We further partition the set of continuous variables in three disjoint sets to represent the partitioning of variables in a system of DAEs (see Section 3.2.2):

  - *X is the set of* state variables*;*
  - *U is the set of* input variables*; and*
  - *Y is the set of* algebraic variables*. The effort and flow variables of the terminals belong to the algebraic variables.*
- *$T_{\mathcal{C}} \subseteq Y \times Y$ is the set of* terminals *of the components.*

  An element $(e, f) \in T_{\mathcal{C}}$ represents a terminal of the component, where $e \in Y$ is the effort variable and $f \in Y$ is the flow variable. We require each variable to appear at most once in the pair (e, f). In this way, we ensure that each variable is used consistently either as an effort or as a flow variable and that each variable is used in at most a terminal.
- *$Init(B, R)$ is a formula describing the set of possible* initial states *of the component.*

- $Invar(B, R)$ *is a formula describing the* invariant condition.

  Such invariant condition holds for the component during its entire execution.

- $Trans(B, R, B', R')$ *is a formula defining the* transition relation *that describes how an instantaneous transition changes the discrete state and the state variables.*

- $Flow(B, \dot{R}, R)$ *is a formula defining the* flow condition.

  The flow condition defines the constraints on the time derivative $\dot{R}$ of the continuous variables X.

- $Input(U, B)$ *is a formula defining the value of the input variables.*

  We assume $Input(U, B)$ to assign a unique and single value $\vec{r} \in \mathbb{R}^{|U|}$ for every possible discrete state 2^B. That is, we require Input to define a function assigning a constant value to U for every possible discrete configuration of the component[1].

Remark 1 (Non-convexity of Invariant Conditions)**.** *We further require both the conditions Flow and Invar to be convex once fixed a discrete state. That is, the formula obtained from Flow (resp. Invar) after assigning a value to all the Boolean variables B and interpreting the remaining variables as reals must describe a convex set. This condition avoids to have differential inclusions in Flow and complicated encoding of the invariants (see the encoding in [CMT14] for a possible solution to this problem).*

[1] While in our presentation we assign a constant value to each input variable, we can generalize our framework assigning a continuous function of time to each input variable. With such extension we could model, for example, power generators with a sinusoidal input (e.g., $u = \sin(t)$, where t is the continuous variable tracking the amount of time elapsed in the system). The algorithms we present below for validating and reformulating the network still work, without any change, in such extended settings. The verification techniques we use [CGMT15], instead, cannot currently analyze systems containing such time-dependent signals, except when these signals is rewritten in the form of an affine ODE dynamic.

Example 1 (Component description of a capacitor)**.** *We use our formalism to model (the component of) a capacitor with capacitance* 2 *farad.*

The capacitor has a single discrete state that we model with a single Boolan variable m *(*$B := \{m\}$*), further imposing that such configuration never change (*$Trans := (m \leftrightarrow m')$*). The capacitor has two terminals* (e_1, f_1) *and* (e_2, f_2)*, where* e_1*,* e_2*,* f_1*, and* f_2 *are the effort and flow output variables. Our main goal is to model the relationship between the current flowing between the two capacitor's terminals and the voltage between the two terminals. We introduce a continuous variable,* $X := \{v\}$*, representing the voltage on the capacitor, and we specify its relationship with the current flowing between the capacitor's terminals:* $\dot{v} = \frac{1}{2} f_1$ *in the flow condition. In the flow condition, we further specify the other relationships between the state variable and the terminals:* $v = e_1 - e_2 \wedge f_1 + f_2 = 0$*. The definition of the component is thus:*

$$
\begin{aligned}
c := \langle & \{m\}, \\
& \{v, e_1, f_1, e_2, f_2\}, \\
& \{(e_1, f_1), (e_2, f_2)\}, \\
& \top, \\
& \top, \\
& (m \leftrightarrow m' \wedge v = v'), \\
& \dot{v} = \frac{1}{2} f_1 \wedge v = e_1 - e_2 \wedge f_1 + f_2 = 0, \\
& \{\} \rangle
\end{aligned}
$$

Example 2 (Component description of an ideal switch)**.** *We model an ideal switch just introducing a Boolean variable,* m*, and specifying the effect on the component's terminals when the switch is open (i.e.,* m*) and closed (i.e.,* $\neg m$*):*

$$
\begin{aligned}
c := \langle & \{m\}, \\
& \{e_1, f_1, e_2, f_2\}, \\
& \{(e_1, f_1), (e_2, f_2)\}, \\
& \top, \\
& \top, \\
& \top, \\
& f_1 + f_2 = 0 \wedge (m \rightarrow (e_1 = e_2)) \wedge (\neg m \rightarrow (f_1 = 0)), \\
& \{\}\rangle
\end{aligned}
$$

Definition 3 (SMDKN Syntax)**.** *A Switched Multi-Domain Kirchhoff Network* $\mathcal{N} := \langle \mathcal{C}, \mathcal{T} \rangle$ *is composed by a list of components* $\mathcal{C} := c_1, \ldots, c_n$ *and a list of nodes formed by components' terminals* $\mathcal{T} := t_1, \ldots t_k$ *where for all* $t \in \mathcal{T}$, $t \subseteq \bigcup_{c_i} T_{\mathcal{C}i}$.

4.2 Defining the Smdkn Semantic

We define all the possible runs of a SMDKN as a hybrid automaton. We define a hybrid automaton from the definition of the network's components and the network's topology using the Kirchhoff laws. The runs of the network are exactly the runs of such hybrid automaton.

We first use the Kirchhoff laws to define the algebraic constraints $K_{\mathcal{T}}$ for the nodes of the network.

$$
K_{\mathcal{T}} := \left(\bigwedge_{t \in \mathcal{T}} \sum_{(e,f) \in t} f = 0 \right) \wedge \left(\bigwedge_{t \in \mathcal{T}} \bigwedge_{(e_1,f_1) \in t} \bigwedge_{(e_2,f_2) \in t} e_1 = e_2 \right) \tag{4.1}
$$

The first conjunct in the formula $K_{\mathcal{T}}$ represents the conservation of flow for each node in the network (i.e., the sum of all the flows on a node, represented as the flow variables on the terminals, must be zero). The second

conjunct of the formula $K_{\mathcal{T}}$ represents the conservation of the potential in (every loop of) the network. The law imposes that the sum of the potential (e.g., voltage in the electrical domain) in a closed path of the network must be zero. Since the SMDKN represents directly the effort variables for each terminal, the formula $K_{\mathcal{T}}$ equivalently encodes that all the effort variables on the same node are equal.

Definition 4 (Hybrid Automaton of a SMDKN)**.** *The hybrid automaton* $H_{\mathcal{N}} := \langle B, R, Init, Invar, Trans, Flow \rangle$ *of the* SMDKN $\mathcal{N} := \langle \mathcal{C}, \mathcal{T} \rangle$ *is defined as follows:*

- $B := \bigcup_{c \in \mathcal{C}} B_c$
- $R := \bigcup_{c \in \mathcal{C}} R_c$
- $Init(B, R) := \bigwedge_{c \in \mathcal{C}} Init_c$
- $Trans(B, R, B', R') := \bigwedge_{c \in \mathcal{C}} Trans_c$
- $Invar(B, R) := K_{\mathcal{T}} \wedge \bigwedge_{c \in \mathcal{C}} Invar_c \wedge \bigwedge_{c \in \mathcal{C}} Input_c$
- $Flow(B, \dot{R}, R) := \bigwedge_{c \in \mathcal{C}} Flow_c$

The hybrid automaton has the same state space of the network (same set of discrete and continuous variables) and has the same initial conditions.

The hybrid automaton encodes the synchronous composition of the network's components where at every discrete transition all the components perform a transition. The model is not restrictive in that an asynchronous composition can be "encoded" using the synchronous framework, as done for example in [CMT11a] introducing stutter transitions in the components.

We define the continuous evolution of the hybrid automaton in the *Invar* and *Flow* constraints. The *Flow* constraint considers all the flow conditions specified in each component — hence, the continuous variables follow the set of trajectories specified locally in each components. However,

the trajectories further depend on the invariant condition on each component, which holds continuously in time, the input value in each component's state, and the composition of the components that the network's topology induces. The hybrid automaton considers such constraints in its invariant definition ($Invar$). A "peculiar" characteristic of Kirchhoff Networks is that the flow condition in a component defines a differential algebraic equation instead of an ordinary differential equation. As an example, consider the component definition in Example 1: the flow condition contains the constraint $\frac{1}{2}f_1$, containing the continuous variable f_1, but it does not define $\dot{f}_1$. Instead, we may determine if there exists a derivative for f_1, and hence an ordinary differential equation, only after we compose the component with the rest of the network.

Definition 5 (SMDKN Semantic)**.** *A* SMDKN $\mathcal{N}$ *accepts all and only the runs accepted by the hybrid automaton* $H_{\mathcal{N}}$*.*

Chapter 5

Validation and Reformulation problems

5.1 Validation

Let a network $\mathcal{N} = \langle \mathcal{C}, \mathcal{T} \rangle$ be given. Our first goal is to automatically check if $\mathcal{N}$ contains inconsistencies, which represent an unwanted condition in the real system modeled by the network.

Definition 6. *A mode m of a network $\mathcal{N}$ is* consistent *if, for every possible assignment to the state (X) and input (U) variables, the linear system* DAE(m) *has at least a solution.*

An inconsistent mode in the network represents an undesired condition in the physical system that must be avoided. Recalling the electrical network of Figure 2.1, the discrete mode M_1 of the network is inconsistent because the current source I_S tries to produce a current not allowed by the the network because the three switches S_0, S_1, and S_2 are open. This inconsistency is hidden in the DAE constraints of the mode M_1. Clearly, inconsistent modes in the design are undesirable, since the behavior of the real system would violate some physical laws. Thus, checking if a mode is consistent is a fundamental step in the validation of $\mathcal{N}$.

Our second goal is to verify safety properties on $\mathcal{N}$. In order to leverage existing tools for the verification of hybrid systems (e.g. HyCOMP [CGMT15]), we need to express the continuous dynamics of $\mathcal{N}$ as ODEs. This means, for every discrete mode of the network, being able to rewrite the DAE Dae(m) as

$$\dot{\vec{X}} = \vec{A} \cdot \vec{X} + \vec{B} \cdot \vec{U}$$

where $\vec{A} \in \mathbb{R}^{|X| \times |X|}$, and $\vec{B} \in \mathbb{R}^{|X| \times |U|}$. This amounts to find a function that, for every possible value of the state variables X and input variables U, returns a value for the first derivative variables $\dot{X}$. The existence of the ODE function requires two necessary conditions: consistency, and determinicity of the values assigned to $\dot{X}$.

Definition 7. *A mode m of a network $\mathcal{N}$ is* deterministic *if, for every possible value of the state X and input variables U, the linear system* Dae(m) *admits at most one solution of the first derivative $\dot{X}$.*

In the example of Figure 2.1, the DAE of the discrete modes M_4 and M_7 contain a capacitor-loop that make the modes inconsistent. Additionally, this discrete modes are also non-deterministic in terms of the first-derivative $\frac{dV_{C_1}}{dt}$ and $\frac{dV_{C_2}}{dt}$ because the currents $C_1.i_+$ and $C_2.i_+$ through the capacitors cannot be uniquely determined due to the capacitor-loop. Thus, the ODE reformulation cannot be computed in the mode M_4 ad M_7.

Definition 8. *A mode m of a network $\mathcal{N}$ is* valid *if it is both* consistent *and* deterministic*. The network $\mathcal{N}$ is* valid *if all the modes $m \in 2^B$ are valid.*

Definition 9 (Validation problem)**.** *Given a network $\mathcal{N}$, the validation problem consists of deciding if $\mathcal{N}$ is valid.*

Remark 2. *According to this definition of validity, we note that a mode is valid if it is associated to a* index-1 *DAE, while it is invalid for* higher-index *DAEs.*

5.2 Reformulation

Definition 10 (Reformulation problem)**.** *Given a valid network $\mathcal{N}$, the* reformulation problem *consists of obtaining a hybrid automaton H with* ***ODE dynamics*** *that is equivalent to $H_{\mathcal{N}}$.*

The reformulated automaton represents the same discrete modes as the network $\mathcal{N}$, but its continuous dynamics is expressed as a system of ODEs. Such representation exists since the network is valid.

Chapter 6

Validation and Reformulation of SMDKN

6.1 Basic Validation and Reformulation

Our technique performs the following steps to produce a symbolic hybrid automaton $H_{\mathcal{N}}$ amenable to verification from the network $\mathcal{N} = \langle \mathcal{C}, \mathcal{T} \rangle$:

1. Check if all the modes of $\mathcal{N}$ are consistent. If it is the case, we proceed to the next step.
2. Check if all the modes of $\mathcal{N}$ are deterministic. If it is the case, $\mathcal{N}$ is valid and we proceed to the reformulation.
3. Reformulate all the modes of $\mathcal{N}$ and define $H_{\mathcal{N}}$.

In the case $\mathcal{N}$ is not consistent, our approach finds all the non-consistent modes, that can be used by the designer to fix the network. While we restrict the presentation to the case where $\mathcal{N}$ is consistent, our approach also performs a *partial reformulation*, that reformulates the DAEs only for the consistent modes. The partial network is necessary in the common scenario where a discrete controller is composed with the network with the goal of keeping the network *outside* the non-consistent states. In this scenario, our approach allows us to verify if such controller is correct.

Validation and reformulation steps can be done for each mode $m \in 2^B$ of $\mathcal{N}$. However, this is not feasible since the number of modes is exponential in the number of the discrete variables of $\mathcal{N}$. To scale and analyze real networks, we use a symbolic approach. In this section, we present a symbolic validation and reformulation for multi-domain networks. The idea is to express the validation and reformulation problems as a first-order logic formula.

6.1.1 SMT encodings of the network DAEs

We represent all the DAEs of the network $\mathcal{N}$ as the quantifier-free formula:

$$\psi_{\text{DAE}}(B, X, U, Y, \dot{X}) := \bigwedge_{c \in \mathcal{C}} Flow_c \wedge K_{\mathcal{T}} \tag{6.1}$$

ψ_{DAE} predicates over the same variables of the network, so we reuse the same notation introduced in Chapter 4 for the network variables, and contains the Boolean variables B, and the Real variables $X, U, Y, \dot{X}$. The validation and reformulation problems only consider the algebraic relationships among the variables defined by the DAE, while they disregard their dependence on time. Thus, the derivative variables in $\dot{X}$ are treated as Real, and not Continuous, variables. Note that the provided encoding enumerates the network components in place of the network global modes that are exponential in the components, thus preventing the exponential blow-up of the formula ψ_{DAE}.

Lemma 1. *μ is a satisfying assignment of ψ_{DAE} iff $\mu_{|R}$ is a solution of* $\text{DAE}(\mu_{|B})$*, where* $R := X \cup U \cup Y \cup \dot{X}$ □

6.1.2 Checking the network for consistency

All the modes of $\mathcal{N}$ are consistent iff the following formula is valid:

$$\psi_{con}(B) := \forall X, U. \exists Y, \dot{X}. \psi_{\text{DAE}}(B, X, U, Y, \dot{X})$$

ψ_{con} represents the set of all the consistent modes.

6.1.3 Checking the network for determinicity

All the modes of $\mathcal{N}$ are deterministic iff the following formula is valid:

$$\begin{aligned}\psi_{det}(B) := \forall X, U, \dot{X}_1, \dot{X}_2.\\ ((\exists Y.\psi_{\text{DAE}}(B, X, U, Y, \dot{X}_1) \wedge\\ \exists Y.\psi_{\text{DAE}}(B, X, U, Y, \dot{X}_2)) \rightarrow \dot{X}_1 = \dot{X}_2)\end{aligned}$$

ψ_{det} represents the set of all the deterministic modes.

6.1.4 Reformulating the network

We reformulate a valid network $\mathcal{N}$ into the hybrid automaton $H^r_{\mathcal{N}} = \langle B^r, R^r, Init^r, Invar^r, Trans^r, Flow^r \rangle$. $H^r_{\mathcal{N}}$ is defined as the hybrid automaton $H_{\mathcal{N}}$ in the Definition 4, except for $Invar^r$ and $Flow^r$. The invariant condition is given by $Invar^r := \psi_Y \wedge Invar(B, R)$, while $Flow^r := \psi_{\dot{X}}$. The formula $\psi_{\dot{X}}$ is the ODE reformulation of the variables $\dot{X}$, while ψ_Y is a relation that represents the values of the algebraic variables Y w.r.t. the state X and input U variables. While we can compute the relation for ψ_Y as $\exists \dot{X}.\psi_{\text{DAE}}(B, X, U, Y, \dot{X})$, finding the $\psi_{\dot{X}}$ is a more difficult task that requires to solve a quantified formula expressed with non-linear arithmetic terms (that synthesize the coefficients of the ODE). We know that such formula cannot be solved efficiently. We do not try to compute it and in our experiments we try to compute $\exists \dot{X}.\psi_{\text{DAE}}(B, X, U, Y, \dot{X})$. This formula *does not* reformulate the system into an ODE, but the time necessary to solve it provides a lower bound for a more complex formula (i.e. with more quantifiers and over non-linear arithmetic predicates).

6.2 Optimized Validation and Reformulation

We improve the basic validation and reformulation procedures by applying an extension of the *implicit function theorem* [Axl97]. Given a system of linear equalities, the theorem gives the necessary and sufficient conditions that allow us to express the values of a subset of the system variables (the dependent variables) as a function of the remaining variables (the independent variables). For our application, the linear system is the DAE of a mode, the dependent variables are the first-derivatives $\dot{X}$, and the independent variables are the state X and input U. Our problem is slightly more complex, since the DAE also contains the algebraic variables Y. One option is to consider them as dependent variables, requiring to find a function that expresses the value of all the variables in Y. However, this limits the applicability of our approach: while we have to express $\dot{X}$ as a system of ODEs, the underlying verification tool does not impose any restriction on the algebraic variables Y that, for example, can assume a value non-deterministically. For this reason we extend the implicit function theorem as follows.

6.2.1 Implicit Function Theorem

Theorem 1 (Implicit Function Theorem)**.** *Let m, n, l be positive integers. Let $F : \mathbb{R}^{m+n} \to \mathbb{R}^l$ be a homogeneous implicit linear function $F(\vec{w}, \vec{z}) := \vec{A}\vec{w} + \vec{B}\vec{z} = \vec{0}$, where $\vec{w} \in \mathbb{R}^{m\times 1}$, $\vec{z} \in \mathbb{R}^{n\times 1}$, $\vec{A} \in \mathbb{R}^{l\times m}$, and $\vec{B} \in \mathbb{R}^{l\times n}$. Let $\vec{b_i}$ be the i-th column vector of the matrix $\vec{B}$, where $i \in \{1, ..., n\}$. Let w_j be the j-th variable of $\vec{w}$, where $j \in \{1, ..., m\}$. The following two conditions hold:*

1. *consistency condition: for all $1 \leq i \leq n$, the linear system $\vec{A}\vec{w} = \vec{b_i}$ is solvable, and*

2. *determinicity condition: the linear system* $\vec{A}\vec{w} = \vec{0}$ *does not admit any homogeneous solution* $\bar{\vec{w}}_h$ *such that its* j*-th component* w_j *is different from zero*

iff there exists a unique linear function $f_j : \mathbb{R}^n \to \mathbb{R}^1$ *such that* $w_j = f_j(\vec{z})$ *and* $F(w_1, ..., f_j(\vec{z}), ..., w_m, \vec{z}) = \vec{0}$. □

The condition (1) guarantees that the system $\vec{A}\vec{w} = -\vec{B}\vec{z}$ admits at least one solution $\bar{\vec{w}}$ for every assignment to the variables $\vec{z}$, reducing the problem to a *finite* number of n checks; the condition (2) guarantees that, for every assignment to the variables $\vec{z}$, every solution $\bar{\vec{w}}$ admits a unique assignment to its j-th component w_j.

Consider the DAE DAE(m) of the mode m and its matrix representation $\vec{M}\,\dot{\vec{x}} + \vec{N}\,\vec{x} + \vec{O}\,\vec{y} + \vec{P}\,\vec{u} = \vec{0}$ (see Equation 3.1). One can directly apply Theorem 1, just by noticing that DAE(m) is indeed a linear homogeneous implicit function $F(\vec{w}, \vec{z})$, where $\vec{w} := \begin{bmatrix}\dot{\vec{x}} \\ \vec{y}\end{bmatrix}$, $\vec{z} := \begin{bmatrix}\vec{u} \\ \vec{x}\end{bmatrix}$, $\vec{A} := \begin{bmatrix}\vec{M} & \vec{O}\end{bmatrix}$, and $\vec{B} := \begin{bmatrix}\vec{P} & \vec{N}\end{bmatrix}$. If the first condition of Theorem 1 holds for all the columns $\vec{b}_i$ of the concatenated coefficient matrix $\vec{B} := \begin{bmatrix}\vec{P} & \vec{N}\end{bmatrix}$, then the discrete mode m is consistent, while if the second condition holds for all $\dot{x} \in \dot{X}$, then discrete mode m is deterministic. Then, if both conditions hold, the discrete mode m is valid.

6.2.2 Validation

Our goal is to check the validity of the network avoiding the universal quantification on the state and input variables introduced in the formulas in Section 6.1. We achieve this by directly checking the conditions of Theorem 1. The consistency condition (1) of the Theorem 1 is encoded as:

$$\psi_{con}(B) := \bigwedge_{z_i \in U \cup X} \exists \dot{X}, Y. \left(\psi_{\text{DAE}}\left[\delta_{z_i}^{\vec{U}\cdot\vec{X}} / \vec{U}\cdot\vec{X}\right]\right)$$

where $\delta_{z_i}^{\vec{U}\cdot\vec{X}}$ represents the vector of size $|\begin{bmatrix}\vec{u}\\ \vec{x}\end{bmatrix}|$, whose elements are identically zero except for the one corresponding to position of the variable z_i. The formula $\psi_{con}(B)$ represents all the consistent modes. The determinicity condition (2) is encoded in the formula:

$$\psi_{det}(B) := \neg\exists \dot{X}, Y.\left(\psi_{\mathrm{DAE}}\left[\vec{0}/\vec{U}\right]\left[\vec{0}/\vec{X}\right] \wedge \left(\dot{\vec{X}} \neq \vec{0}\right)\right)$$

The formula $\psi_{det}(B)$ represents all the deterministic modes of $\mathcal{N}$. We notice that the effect on ψ_{DAE} of the X and U substitutions is equivalent to symbolically "turning on and off" a subset of the columns of the coefficient matrix $\vec{B} := \left[\vec{P} \;\; \vec{N}\right]$ in order to symbolically check the conditions of the Theorem 1.

Lemma 2. *A network $\mathcal{N}$ is consistent iff for all $m \in 2^B$, $m \models \psi_{con}(B)$, and is deterministic iff for all the modes $m \in 2^B$, $m \models \psi_{det}(B)$* □

6.2.3 Reformulation

The algorithm PerVariableRef (Figure 6.1) synthesizes the formulas $\psi_{\dot{X}}$ and ψ_Y used in the reformulated automaton $H^r_{\mathcal{N}}$, by using Theorem 1 and Lemma 4.

In the algorithm, we use the SMT solver primitives *push*, *assert*, *isSat*, *pop*, *reset* (see e.g. [CGSS13]), *getModel*, to get a satisfying assignment to all the free variables of the formula, and *quantify* to eliminate the quantifiers from the formula.

PerVariableRef invokes the Reform procedure (Figure 6.2) on each variable $\dot{x} \in \dot{X}$ (Line 3), computing the reformulation $Ref_{\dot{x}}$ of the variable $\dot{x}$ and the formula $\psi_{Y,\dot{x}}$. In the algorithm, we compute ψ_Y by directly substituting in ψ_{DAE} the variables $\dot{X}$ with their reformulated value. Since the reformulation of a variable $\dot{x} \in \dot{X}$ depends on the discrete modes, we store this value in a variable $\dot{x}_s$ (we add a the set of variables $\dot{X}_s =$

PERVARIABLEREF (ψ_{DAE}, X, U):
1. $(\psi_{\dot{X}}, \psi_Y) := (\top, \top)$
2. **for each** $\dot{x} \in \dot{X}$:
3. $\quad (Ref_{\dot{x}}, \psi_{Y,\dot{x}}) :=$ REFORM (ψ_{DAE}, X, U, $\dot{x}$)
4. $\quad \psi_{\dot{X}} := \psi_{\dot{X}} \wedge Ref_{\dot{x}}$
5. $\quad \psi_Y := \psi_Y \wedge \psi_{Y,\dot{x}}$
6. $\psi_Y := \psi_Y \wedge \psi_{\text{DAE}}[\dot{X}_s/\dot{X}]$
7. **return** $(\psi_{\dot{X}}, \psi_Y)$

Figure 6.1: Reformulation algorithm for $\mathcal{N}$.

$\{\dot{x}_s \mid \dot{x} \in \dot{X}\}$). $\psi_{Y,\dot{x}}$ represents the values that $\dot{X}_s$ takes depending on the discrete state of the network. At Line 6, the algorithm constructs ψ_Y, that encodes the reformulation values for $\dot{X}_s$ and the ψ_{DAE} formula where all the $\dot{X}$ variables have been substituted with the $\dot{X}_s$ variables.

REFORM works under the *validity* assumption, that ensures the existence of a reformulation, and uses the linearity Lemma 4 to synthesize the reformulation. According to Lemma 4, we know that, for a mode $m \in 2^B$ and a variable $\dot{x}$, the function $f_{\dot{x}}(\begin{bmatrix}\vec{u}\\ \vec{x}\end{bmatrix})$ such that $\dot{x} = f_{\dot{x}}(\begin{bmatrix}\vec{u}\\ \vec{x}\end{bmatrix})$ is defined as $f_{\dot{x}}(\begin{bmatrix}\vec{u}\\ \vec{x}\end{bmatrix}) := \bar{\vec{w}}^j_{p_1} z_1 + ... + \bar{\vec{w}}^j_{p_n} z_n$, where j is the index corresponding to the variable $\dot{x}$ in the vector $\vec{w} := \begin{bmatrix}\dot{\vec{x}}\\ \vec{y}\end{bmatrix}$, and $\bar{\vec{w}}^j_{p_i}$ is the element corresponding to $\dot{x}$ in the i-th particular solution $\bar{\vec{w}}_{p_i}$. Thus, we can synthesize the coefficients of the function $f_{\dot{x}}(\begin{bmatrix}\vec{u}\\ \vec{x}\end{bmatrix})$ by computing all the n *particular* solutions of the system and taking their j-th element.

Figure 6.2 shows the reformulation procedure for a single variable $\dot{x}$: each execution of the loop at Line 4 finds a mode $m \in 2^B$ (Line 5) for which the $\dot{x}$ reformulation is still unknown. Then (Line 6) the algorithm

```
REFORM (ψ_DAE, X, U, ẋ):
1. Ref_ẋ := ⊤
2. ψ_{Y,ẋ} := ⊤
3. solver.assert(⊤)
4. while solver.isSat():
       # Get a fresh mode
5.     m := solver.getModel()
       # Get the row vector of coeff. that
       # contributes to ẋ in m
6.     D := GETCOEFF (ψ_DAE, X, U, ẋ, m)
       # Get the cluster of modes that share the
       # same coeff.
7.     β := GETEQMOD (ψ_DAE, X, U, ẋ, D)
       # Prune the cluster of modes from the search
8.     solver.assert(¬β)
       # Build the reformulation equation
9.     Eq := ẋ = D [u⃗; x⃗]
10.    Ref_ẋ := Ref_ẋ ∧ (β → Eq)
11.    ψ_{Y,ẋ} := ψ_{Y,ẋ} ∧ β → ẋ_s = D [u⃗; x⃗]
12. return (Ref_ẋ, ψ_{Y,ẋ})
```

Figure 6.2: Reformulation of a single variable $\dot{x}$.

computes the coefficients D of the $\dot{x}$ reformulation in m. The procedure computes (Line 7) the cluster β of **all the modes** that share the same coefficients D, and hence the same reformulation, for $\dot{x}$. At Line 8, we prune the search space removing β. Eq is created (Line 9) by computing the product of the coefficients row vector D and the variables column vector $\begin{bmatrix}\vec{u}\\ \vec{x}\end{bmatrix}$.

At Line 10, we accumulate the reformulation (one for each cluster) in the returned formula $Ref_{\dot{x}}$. At Line 11, we construct $\psi_{Y,\dot{x}}$ that constraints

the values of the additional variable $\dot{x}_s$. REFORM terminates when the reformulation of $\dot{x}$ is known for all the modes $m \in 2^B$.

```
GETCOEFF (ψ_DAE, X, U, ẋ, m):
    # D row vector of coeff. w.r.t. U ∪ X
1. coeffSolver.assert(ψ_DAE ∧ m)
2. for each z_i ∈ U ∪ X:
3.     coeffSolver.push()
         # build the rhs corresponding to z_i
4.     rhs_{z_i} := z_i = 1 ∧ ⋀_{l∈(U∪X)\{z_i}} l = 0
5.     coeffSolver.assert(rhs_{z_i})
         # get a system solution
6.     μ' := coeffSolver.getModel()
         # μ'(ẋ) is the coeff. w.r.t. z_i
7.     D[i] := μ'(ẋ)
8.     coeffSolver.pop()
9. return D
```

Figure 6.3: Computes the coefficients D of the reformulation of $\dot{x}$.

GETCOEFF is shown in Figure 6.3. For each variable $z_i \in U \cup X$, the condition built at Line 4 reduces the matrix product $\vec{B}\begin{bmatrix}\vec{u}\\ \vec{x}\end{bmatrix}$ of the ψ_{DAE} formula to the column vector $\vec{b}_i z_i = \vec{b}_i 1 = \vec{b}_i$ that corresponds to the i-th iteration. This formula is asserted in the solver at Line 5. At Line 6, the algorithm finds a *particular* solution $\mu^{'}$ to the system $\vec{A}\vec{w} = -\vec{b}_i$. Then (Line 7) we assign the value $\mu^{'}(\dot{x})$ of the $\dot{x}$ element of the solution $\mu^{'}$ to the i-th reformulation coefficient $D[i]$.

The procedure GETEQMOD (Figure 6.4) builds the condition γ that is satisfiable in every $m \in 2^B$ that shares the same reformulation coefficients for $\dot{x}$. In Line 7, we symbolically compute the set of equivalent modes β.

Theorem 2 (Correctness of the reformulation)**.** *Given a valid network $\mathcal{N}$, the hybrid automaton $H^r_{\mathcal{N}}$ is equivalent to the hybrid automaton $H_{\mathcal{N}}$ that*

GetEqMod (ψ_{DAE}, X, U, $\dot{x}$, D):
1. eqSolver.reset()
2. $\gamma := \top$
3. **for each** $z_i \in U \cup X$:
4. $\quad rhs_{z_i} := z_i = 1 \wedge \bigwedge_{l \in (U \cup X) \setminus \{z_i\}} l = 0$
5. $\quad \gamma_{z_i} := \dot{x} = D[i] \wedge rhs_{z_i}$
6. $\quad \gamma := \gamma \wedge \exists X, U, Y, \dot{X}.(\psi_{\text{DAE}} \wedge \gamma_{z_i})$
7. $\beta :=$ eqSolver.quantify(γ)
8. **return** β

Figure 6.4: Find the cluster of modes that share the same coefficients D for $\dot{x}$.

defines the network semantics. □

Chapter 7

Experimental evaluation

This experimental evaluation compares the performance of the proposed encodings on different scenarios. The algorithms are compared in terms of their execution time and scalability on several scalable benchmarks taken from literature and industrial applications. First, we compare the validation algorithms to analyze both valid and invalid networks. Second, we apply the reformulation algorithms to the network, restricting the reformulation to the valid modes of the networks that are globally not-valid. Last, we model-check safety properties on the reformulated networks to demonstrate the practical applicability of the proposed SMT framework.

7.1 Setup

We implemented the symbolic validation and reformulation approaches using the pySMT [GM15] python library and the MathSAT5 [CGSS13] SMT-solver. At the verification core, we use the symbolic model checker HyCOMP [CGMT15]. Our work-flow takes as input a Smdkn and a safety property, and performs *validation* (Val), *reformulation* (Ref), and *verification* (Ver). The validation and the reformulation come with two variants, basic (Bas), and optimized (Opt). Bas refers to the algorithms of Section 6.1, Opt refers to those of Section 6.2.

We run the experiments on a 3.5 GHz cpu with 16GB RAM, with time out (TO) set to 7200s for VAL and REF, and 18000 s for VER. The tools and the benchmarks are available at `http://es.fbk.eu/people/sessa/attachment/phdthesis/phdthesis.tar.bz2`.

7.2 Benchmarks

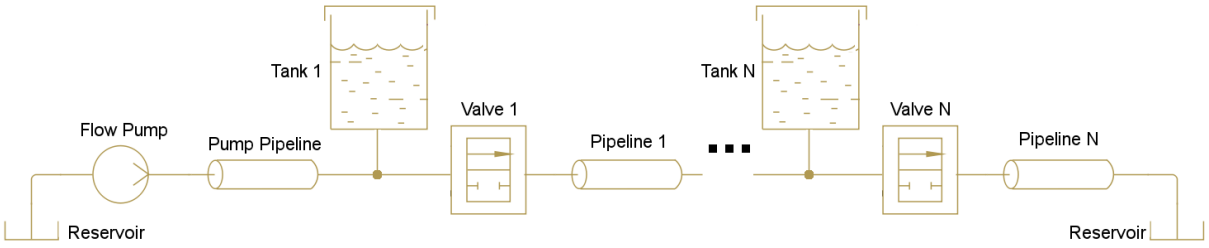
We consider six scalable benchmarks chosen to stress the main aspects of our approach in the validation, reformulation, and verification tasks. The benchmarks come from four families of systems: the Water Tank System (WTS), the Non-Linear Transmission Line (NLTL), the Wheel Braking System (WBS), the Landing Gear System (LGS). Table 7.1 summarizes the characteristics of the benchmark families.

	WTS	NLTL	WBS	LGS
Valid	No	Yes	Yes	Yes
Dynamics	ODE	ODE	ODE	PWL

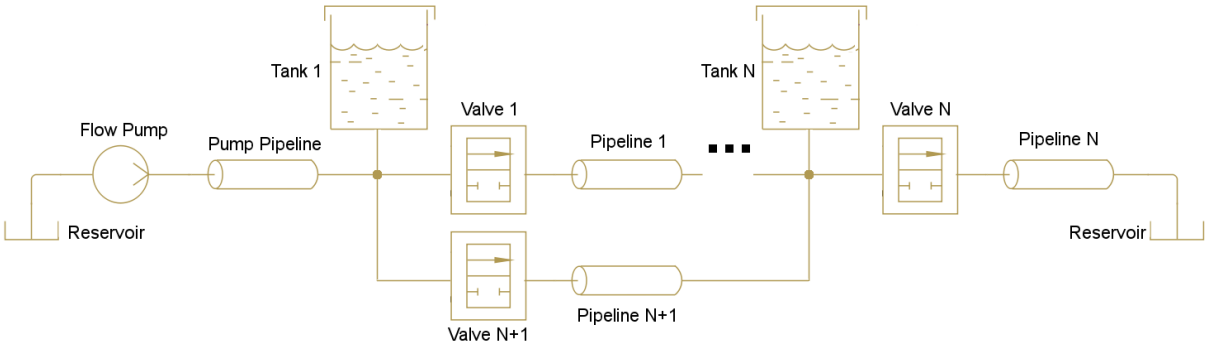
Table 7.1: Main properties of the benchmarks.

The Water Tank System (WTS) is a set of monolithic hydraulic benchmarks. The WTSLIN$_{[N]}$ and WTSRING$_{[N]}$ benchmarks of Figure 7.1 represent networks of N hydraulic tanks connected at the base and subjected to the same atmospheric pressure. The connection channels between tanks are composed of valves and pipelines. The connection topology is linear in WTSLIN$_{[N]}$ (Figure 7.1a) and circular in WTSRING$_{[N]}$ (Figure 7.1b). The flow pump on the left provides water to the tanks, while the flush valve on the right discharges water from the system. When a channel between two tanks is open, the water bidirectionally flow across the tanks according to the communicating vessels principles. A full tank cannot accept further incoming fluid. This is not an harmful condition by itself, but it may lead to a pump failure if the pump is connected to a full tank and to a closed

channel. This is an hazardous condition for the entire system and we will detect it discovering the invalid modes of the models. The WTS benchmarks are originally proposed in [BBGJ15], with a hand-crafted technique meant for the automatic generation of hybrid benchmarks that abstracts away the mutual interactions among the water levels stored in the tanks. On the contrary, our work aims at faithfully representing the physics of the real system, retaining the compositional structure of the physical system.



(a) Linear topology ($\text{WTSLIN}_{[N]}$)



(b) Ring topology ($\text{WTSRING}_{[N]}$)

Figure 7.1: Water tanks system with N tanks.

The Non-Linear Transmission line (NLTL) benchmarks of Figure 7.2 is commonly used in the engineering practice to represents the propagation of an electromagnetic signal along a non-linear transmission line in the form of a an electrical lumped element model. We recall that a lumped element model simplifies the description of the behavior of spatially distributed physical systems (eg the non-linear transmission line) into a topology consisting of discrete entities that approximate the behavior of the distributed

system.

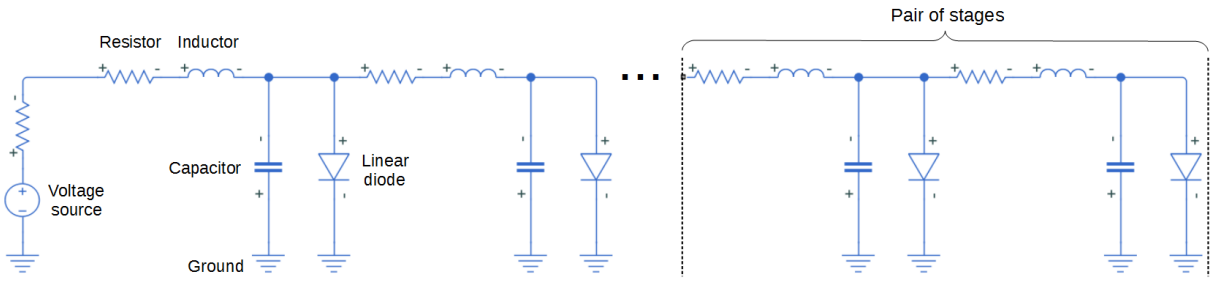
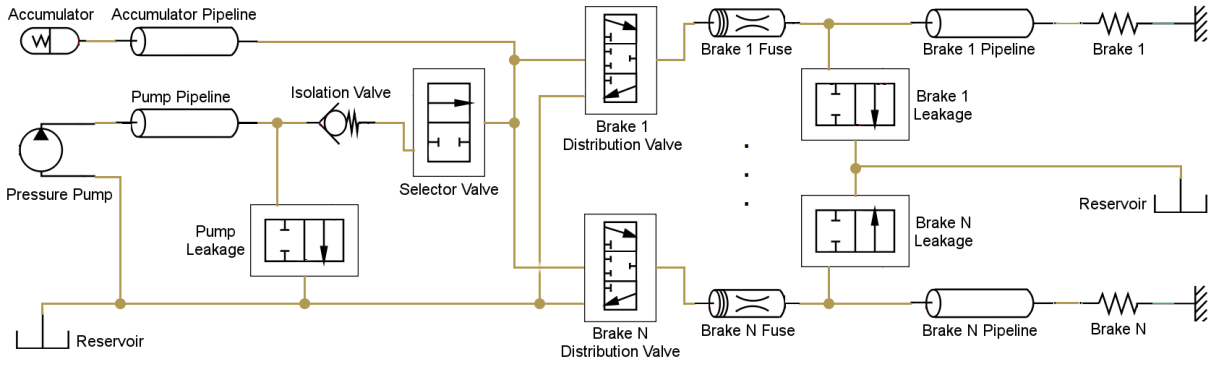


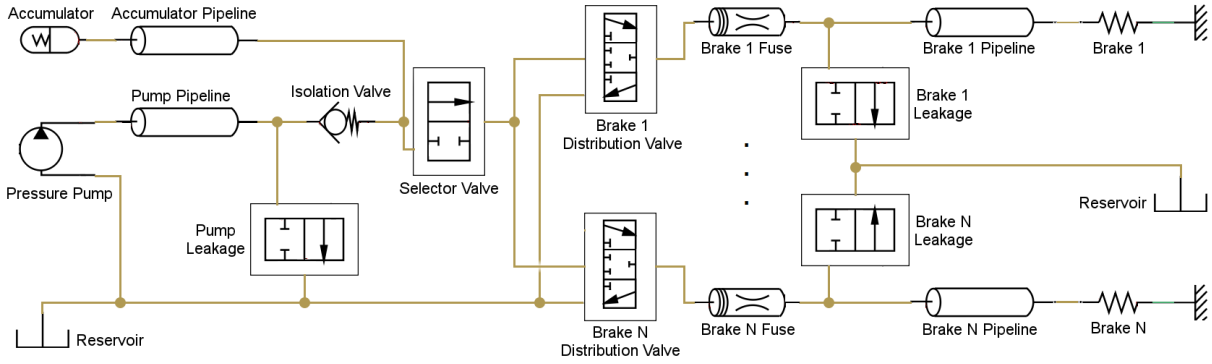
Figure 7.2: Lumped element model of the Non-Linear Transmission Line with N pairs of stages ($\text{NLTL}_{[N]}$).

The Wheel Braking System (WBS) discussed in the SAE standard AIR6110 [SAE11] is a monolithic power-redundant hydro-mechanical system designed to stop an aircraft. The two architectures $\text{WBSA2}_{[N]}$ and $\text{WBSA4}_{[N]}$ of Figure 7.3 dispatch energy either from the main pressure pump or the redundant accumulator on the left to the N wheel braking lines on the right through a network of hydraulic pipelines and valves. In turn, the brakes discharge their energy towards the fluid reservoir. The selector valve disables the braking function by disconnecting the braking lines from the power units. $\text{WBSA2}_{[N]}$ and $\text{WBSA4}_{[N]}$ differ for the insertion point of the accumulator line: $\text{WBSA2}_{[N]}$ contains a design flaw, fixed in $\text{WBSA4}_{[N]}$, because the accumulator is connected downstream of the selector valve, and this may inadvertently activate the brakes even if the selector valve is closed. The models consider several pipeline faults in the form of fluid leakages. First, when the pipeline of the main pump fails, the redundant hydraulic accumulator still supplies energy to the braking lines while the isolation valve prevents the accumulator to discharge through the broken pipeline. Second, when the pipeline of a braking line fails the corresponding hydraulic fuse detects and stops the fluid leakage, isolating the faulty brake. Since our modeling framework preserves the compositionality of the original systems, we highlight that a fault-free model is easily

fault-extended by connecting additional faulty components to the nominal model. Such components introduce into the model new discrete modes with the physics necessary to describe the faults, e.g. the fluid leakages of our WBS benchmarks.



(a) Arch.2 ($\text{WBSA2}_{[N]}$)



(b) Arch.4 ($\text{WBSA4}_{[N]}$)

Figure 7.3: Wheel Braking System with N braking lines

The Landing Gear System (LGS) of Figure 1.1 is a monolithic hydro-mechanical system that control the undercarriage of an aircraft. The flow pump operates two series of hydraulic cylinders that in turn drive mechanical loads along a straight line. For every series of cylinders, a distribution valve determines the moving direction of the cylinder rod by routing the fluid through the two chambers of the cylinders. When the hydraulic pressure in the left chamber is greater than the pressure in the right chamber, the rod moves from left to right. Similarly in the other direction. A shut-

off valve disconnects the pressure pump from the cylinders in order to lock down the rod positions. LGS$_{[N]}$ is inspired to [BW14], and is parameterized with respect to the number N of hydraulic cylinder lines.

	Boolean vars					Real vars				
N	2	4	6	8	10	2	4	6	8	10
WTSLIN$_{[N]}$	4	8	12	16	20	58	100	142	184	226
WTSRING$_{[N]}$	5	9	13	17	21	70	112	154	196	238
NLTL$_{[N]}$	4	8	12	16	20	70	126	182	238	294
WBSA2$_{[N]}$	11	19	27	35	43	119	185	251	317	383
WBSA4$_{[N]}$	11	19	27	35	43	119	185	251	317	383
LGS$_{[N]}$	9	13	17	21	25	94	142	190	238	286

Table 7.2: Size of the encoding of the benchmarks in terms of Boolean and Real variables.

Table 7.2 and Table 7.3 report the size of the SMT encoding of the benchmarks in terms of the Boolean and Real-valued variables needed to encode the discrete modes and physical parts of the models, respectively. Clearly, the size of the state-space makes manual inspection extremely time-consuming, expensive, and unfeasible in practice.

Most of the benchmarks considered in this evaluation cannot be analyzed with the approach presented in [CMS16]. There are several reasons for this. First, the benchmarks different from the NLTL are out of the electrical domain. Even if [CMS16] deals with some simple hydraulic components by means of the hydraulic-electrical analogy, the cylinder component used in the LGS does not fit in the domain analogy. Second, [CMS16] cannot deal with non-deterministic algebraic variables. Our WBS benchmarks yield under-specified algebraic variables that were not present in the much simpler and less complete model used in [CMS16]. Also, note that our modeling of the WBS benchmarks is different than the model presented in [Bea15], which is an abstract, discretized and causal model of the system suitable to perform a formal system safety assessment analysis in a single

		2	3	4	5	6	7	8	9	10
WtsLin$_{[N]}$	Input	1	1	1	1	1	1	1	1	1
	State	2	3	4	5	6	7	8	9	10
	Output	53	72	91	110	129	148	167	186	205
WtsRing$_{[N]}$	Input	1	1	1	1	1	1	1	1	1
	State	2	3	4	5	6	7	8	9	10
	Output	65	84	103	122	141	160	179	198	217
Nltl$_{[N]}$	Input	1	1	1	1	1	1	1	1	1
	State	8	12	16	20	24	28	32	36	40
	Output	101	145	189	233	277	321	365	409	453
WbsA2$_{[N]}$	Input	1	1	1	1	1	1	1	1	1
	State	3	4	5	6	7	8	9	10	11
	Output	112	143	174	205	236	267	298	329	360
WbsA4$_{[N]}$	Input	1	1	1	1	1	1	1	1	1
	State	3	4	5	6	7	8	9	10	11
	Output	112	143	174	205	236	267	298	329	360
Lgs$_{[N]}$	Input	3	4	5	6	7	8	9	10	11
	State	2	3	4	5	6	7	8	9	10
	Output	87	108	129	150	171	192	213	234	255

Table 7.3: Size of the encoding of the benchmarks in terms of input/state/output Real variables. The number of the reformulated variables (i.e. the dotted variables) is equal to the state variables.

global discrete mode of the network. Instead, in our Wbs benchmark we model the real continuous physics of the system.

7.3 Validation

The results of the evaluation are summarized in Table 7.4. First, we consider the runtime of the basic (Bas) and the optimized (Opt) encodings for validation. We see that Opt solves 46 out of 54 instances, while Bas times out on the 29 biggest instances. Focusing on the instances solved by both encodings, Opt outperforms Bas by three orders of magnitude

		2	3	4	5	6	7	8	9	10
WtsLin$_{[N]}$	Bas	1	1	3	14	127	1426	to	to	to
	Opt	1	1	1	1	2	4	7	9	15
WtsRing$_{[N]}$	Bas	1	4	47	661	to	to	to	to	to
	Opt	1	1	2	3	4	12	22	38	89
Nltl$_{[N]}$	Bas	1	1	2	12	124	1837	to	to	to
	Opt	1	1	1	1	1	2	2	2	2
WbsA2$_{[N]}$	Bas	10	252	to	to	to	to	to	to	to
	Opt	1	2	20	195	2578	to	to	to	to
WbsA4$_{[N]}$	Bas	9	279	to	to	to	to	to	to	to
	Opt	1	2	15	156	1913	to	to	to	to
Lgs$_{[N]}$	Bas	1	5	26	218	4074	to	to	to	to
	Opt	1	1	2	5	13	37	122	413	1284

Table 7.4: Validation time [s]. (to = timeout)

and scales much better w.r.t the benchmark size. Noteworthy, the Opt method validates 2^{27} discrete modes of the WbsA4$_{[6]}$ instances within 1913 seconds. These results provide a clear evidence that the Bas encodings is infeasible for real life systems, while Opt offers an efficient solution to solve the validation problem.

The Nltl, Wbs, and Lgs benchmarks have only consistent modes. This does not hold for the Wts benchmarks, where a full tank cannot accept further incoming liquid from the pump. We remind that the *full* mode of the tanks may lead to several hazardous configurations of the network, that correspond to all the valve configurations that connect the pump to a sequence of full tanks ended by a closed channel. Our validation approach is able to detect and report such bad configurations.

Although the reported validation runtimes make the proposed SMT-based approach feasible for real-size application, the general trend of the validation runtime appear to increase exponentially with the number of the switching components in the network. Noteworthy, this exponential trend

does not affect the WTS and NLTL benchmarks where the validation time appears independent from the network size. We can make two considerations about the better efficiency of the SMT engine on this benchmarks. The WTS benchmarks are not-valid because they contain several inconsistent modes and the SMT engine is much more efficient in discovering non-valid modes rather than valid modes. On the other side, the structure of the NLTL benchmark contains strong symmetries that maximize the sharing of continuous dynamics, exploiting the symbolic ability of the SMT engines that are really efficient on this kind of encodings. In the following reformulation Section, we provide further details on the sharing of continuous dynamics.

7.4 Reformulation

We consider the runtime for the BAS reformulation, and the OPT reformulation. The reformulation is performed only for the benchmark instances that completed the validation procedure; the reformulation not run are marked with a NA symbol in the results. From the results of Table 7.5, the BAS encoding cannot deal with the benchmarks, whereas the OPT encoding successes in reformulating all the instances that were validated in the previous Section, except for the $\text{WBSA4}_{[6]}$ and $\text{LGS}_{[10]}$ instances that run out of time. Again, this happens because the OPT encoding exploits the properties of the algebraic structure of the problem to mitigate the computational complexity of the quantifier elimination in the computation of the first-derivative reformulations. Additionally, the syntactic substitution of the first-derivative reformulation into the network DAE formula completely avoids the need for the quantifier elimination step in the reformulation of the algebraic variables.

We notice that the reformulation of the WTS benchmarks is restricted

		2	3	4	5	6	7	8	9	10
WtsLin$_{[N]}$	Bas	8	1072	to	to	to	to	na	na	na
	Opt	1	1	3	7	17	34	62	118	213
WtsRing$_{[N]}$	Bas	39	to	to	to	na	na	na	na	na
	Opt	1	2	8	21	52	114	243	498	1732
Nltl$_{[N]}$	Bas	100	to	to	to	to	to	na	na	na
	Opt	3	6	13	22	40	69	104	135	183
WbsA2$_{[N]}$	Bas	to	to	na	na	na	na	na	na	na
	Opt	3	11	97	1072	na	na	na	na	na
WbsA4$_{[N]}$	Bas	to	to	na	na	na	na	na	na	na
	Opt	3	12	89	905	to	na	na	na	na
Lgs$_{[N]}$	Bas	187	to	to	to	to	na	na	na	na
	Opt	2	4	13	46	138	486	1636	5330	to

Table 7.5: Reformulation time [s]. (to = timeout, na = not available because the validation run out of time and the reformulation is not performed).

to the *valid* modes. The ability of representing these non-valid networks is crucial when considering the functional verification of the network composed with a controller designed to prevent the reachability of hazardous configurations.

		2	3	4	5	6	7	8	9	10
WtsLin$_{[N]}$	Total	10	22	40	65	98	140	192	255	330
	Max	6	9	12	16	20	25	30	36	42
	Min	4	5	6	7	8	9	10	11	12
	Avg	5.0	7.3	10.0	13.0	16.3	20.0	24.0	28.3	33.0
WtsRing$_{[N]}$	Total	14	53	110	200	320	483	686	942	1248
	Max	8	20	32	46	62	80	100	122	146
	Min	6	15	23	35	47	63	79	98	117
	Avg	7.0	17.7	27.5	40.0	53.3	69.0	85.7	104.7	124.8
Nltl$_{[N]}$	Total	12	18	24	30	36	42	48	54	60
	Max	2	2	2	2	2	2	2	2	2
	Min	1	1	1	1	1	1	1	1	1
	Avg	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
WbsA2$_{[N]}$	Total	35	98	269	720	NA	NA	NA	NA	NA
	Max	17	41	97	225	NA	NA	NA	NA	NA
	Min	9	19	43	99	NA	NA	NA	NA	NA
	Avg	11.7	24.5	53.8	120.0	NA	NA	NA	NA	NA
WbsA4$_{[N]}$	Total	39	119	345	955	TO	NA	NA	NA	NA
	Max	17	41	97	225	TO	NA	NA	NA	NA
	Min	11	26	62	146	TO	NA	NA	NA	NA
	Avg	13.0	29.7	69.0	159.2	TO	NA	NA	NA	NA
Lgs$_{[N]}$	Total	12	32	88	218	540	1270	2960	6706	TO
	Max	6	12	22	46	90	186	370	754	TO
	Min	6	10	22	42	90	178	370	738	TO
	Avg	6.0	10.7	22.0	43.6	90.0	181.4	370.0	745.1	TO

Table 7.6: Size of the reformulation. Total represents the sum over the reformulated variables of the discovered equivalence classes. Max/Min is the size of the largest/smallest partition in terms of equivalence classes. Avg is the size in terms of equivalence classes of the average partition over the reformulated variables.

Similarly to the validation problem, the general trend of the reformulation time is exponential in the number of switching components. This trend is much less pronounced for the Nltl benchmark. Table 7.6 shows the size of the reformulations in terms of the discovered distinct continuous

dynamics. The average size of a first-derivative reformulation of the NLTL benchmark is **constant** in practice and equal to 1.5 distinct dynamics. In fact, thanks to the symmetries of the NLTL network, every reformulated first-derivative variable experiences **at most two distinct continuous dynamics** regardless of the size of the network. This extreme sharing of continuous dynamics makes the total number of discovered dynamics linear in the number of switching components, and the SMT engine can benefit from this structural property of the network.

7.5 Verification

We model check the WBS and LGS benchmarks with the symbolic model checker HYCOMP [CGMT15]. We use the HYCOMP tool as-is and without any tight integration of our procedure into the model checker source code.

For both WBS benchmarks we consider the safety property "*when the selector valve is closed, a brake command cannot actuate any brake*". Consistently with the SAE standard AIR6110 [SAE11], that describes such design flaw, the safety property is violated for $\text{WBSA2}_{[N]}$ and is verified by $\text{WBSA4}_{[N]}$. For the LGS benchmark, we check the existence of an expected execution scenario with the invariant property "*the first cylinder cannot reach its end-of-stroke*". The property is violated as expected, demonstrating the existence of the execution scenario for the first cylinder.

The verification on the hybrid automata from the OPT reformulation completes within the time out on all the benchmarks, returning the expected results. Finding the violation in $\text{WBSA2}_{[N]}$ is slightly faster than proving the property in $\text{WBSA4}_{[N]}$. Overall, these results provide empirical evidence of the applicability of our approach in the formal verification of real world hybrid system represented as a SMDKN.

Chapter 8

Case study: Railway Relay Interlocking Systems

We present the results of the experimentation of our methodology on a railway case-study developed in collaboration with Italian railway company (Rete Ferroviaria Italiana). The work appeared in [CCM+18].

Figure 8.1: An example of railway system controlled by Ris.

Railway signaling systems guarantee the safe operation of train traffic. Trains run between points of the rail network, moving from section to section along exclusively allocated routes and crossing roads (Figure 8.1).

Protection against catastrophic events, such as train-to-train and train-to-car collisions, is devoted to various devices such as semaphores, barriers at the level crossing, and train detection systems. These devices must be suitably controlled and coordinated by a logic that ensures the safety of operation even in case of multiple device faults.

Traditionally, the logic has been implemented by means of the Relay technology, in the form of networks of interconnected analog electromechanical components, such as power supplies, contacts, circuit breakers, and many forms of electrically-controlled contacts, also known as *relays*. This components are allocated in physical rack similar to those shown in Figure 8.2.

RIS are progressively being replaced by computer-based logics (CBL), that ensure greater flexibility and lower cost. The key question is how to ensure that the CBL is compliant with the (trusted) behavior of the relay-based interlocking being replaced. In some sense, the specification for the CBL is hidden in the relay circuit. Unfortunately, RIS are often old, legacy systems, hard to understand for software engineers at the level of abstraction required to specify the CBL. Thus, the valuable information they encode is not readily available.

Although relays may be thought of as Boolean components, that is just open or closed, this turns out to be a gross simplification. In order to operate (e.g. switching from open to closed), relays may require time, and go through transients required to fully excite the circuitry. Hence, a simple Boolean propagation is in fact a coarse abstraction of a sequence of intermediate states before stability. Furthermore, relays are subject to faults that may either delay or prevent the correct operation. Thus, relay

Figure 8.2: Traditional deployment of realy racks in relay room.

networks are often designed in a redundant fashion in order to mitigate the effect of faults and to ensure safety (at the cost of liveness) in all conditions.

In this thesis, we propose a methodology and a tool chain to analyze and understand legacy RIS, adopted in an ongoing research project of Rete Ferroviaria Italiana (RFI). At the surface, a graphical tool supports the component-based modeling of the RIS. The designer selects components from a palette of over 100 elements, and connects them according to the input description – typically, a printout of the electrical schematic. This step does not require any deep understanding of the nature of the circuit, and ensures that the semantic gap w.r.t. the legacy description is as limited as possible. The corresponding internal representation is reduced to a Switched Multi-Domain Kirchhoff Network (SMDKN), which has a semantic based on Differential Algebraic Equations (DAE). In turn, the SMDKN is compiled into a network of hybrid automata, based on the techniques proposed in [CMS17]. Then, various forms of formal analysis are supported by means of SMT-based model checking. At its core, the approach is based on the modeling of the RIS analog signals (i.e. currents and voltages) over continuous time. The ability to analyze the circuit at the physical level supports a comprehensive understanding at the symbolic level in terms of railways control actions. This is done by defining suitable symbolic predicates in terms of the analog state: for example, a green light to the train may correspond to a suitable current and voltage drop in the corresponding semaphore lamp.

The methodology is fully supported by an automated SMT-based verification tool chain. We evaluated the approach on a set of industrial-size railway RIS, with schematic having more than a thousand components and four-meter long plotter printouts. The results demonstrate practical scalability: we are able to prove (or disprove) conjectured properties, simulate scenarios, and construct fault-trees (FT) corresponding to undesirable

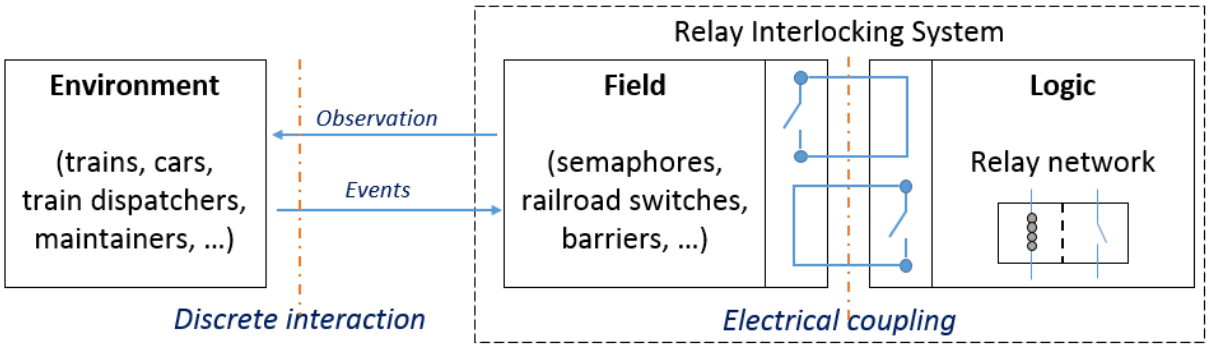


Figure 8.3: Conceptual architecture of a RIS.

events.

This approach was devised as a consequence of a previous unsatisfying modeling attempt we carried on basing our analysis on the traditional formal modeling at the Boolean level. Since relays are not instantaneous Boolean switches, substantial ingenuity from the modeler was required to bridge the gap with respect to the electrical semantics. This made the modeling task unmanageable in terms of conceptual hardness, and led to imprecise results (due to spurious behaviors) that we will report in the following sections. From a pragmatic perspective, the proposed approach provides invaluable support for the understanding of the legacy circuit (and ultimately the reverse-engineering of requirements for the CBL design).

8.1 Relay Interlocking Systems

A Relay Interlocking System (RIS) is an electro-mechanical system that conveys messages between the railway *agents* (e.g., trains, dispatchers, technicians). Figure 8.3 shows the conceptual architecture of a RIS: the agents interacts with the *field devices* (e.g., semaphores, level crossing barriers, railroad switches) that are in turn controlled through the *relay control logic* (an interconnection of relays).

The agents interact with the field devices observing their state (e.g., if a

semaphore light is on or off, the position of a barrier or of a railroad switch) and perform some actions (e.g., toggling an electrical contact, pushing a button) to change the current state of the RIS. The field devices are then connected to the relay control logic that reacts to the state change to implement the signaling system (e.g., lower the barrier of a level crossing when a train is approaching).

The RIS is implemented as a network of switching electro-mechanical components where relays are the main switching components. Relays are electrically-controlled analog switches that implement the relay logic. A relay contains a mechanical contact that can open or close a contact (e.g., a relay can open or close the circuit of a semaphore light turning it on or off). A relay controls its contact with a coil that is physically disconnected from the contact itself. The relay switches the contact when the current that flows in the coil falls within or exceeds a *current threshold*. The relay is in the *dropped state* when the coil's current is below the threshold and it is in the *drawn state* otherwise. When a component in a RIS switches to a different state, for example when an agent pushes a button, it induces different circuit contacts and hence a different behavior of the currents and voltages in the RIS. The changes in the currents and voltages can in turn change the state of the relays in the circuit (e.g., the change of the current on the relay coil switches the state of the relay). Thus, a single state change in the RIS may generate a sequence of subsequent state changes.

A *principle schemata* is the standard graphical representation[1] of the design of a RIS. Figure 8.4 shows the principle schemata for the RIS that controls the semaphore lights for a level crossing (we will refer to this example as R2G1). In the RIS a lever handle (the component named L_1 in the lower left part of the diagram) controls the semaphore for the level

[1]We use the graphical representation defined in the Italian railway regulation UNIFER-CEI S-461 [Com76].

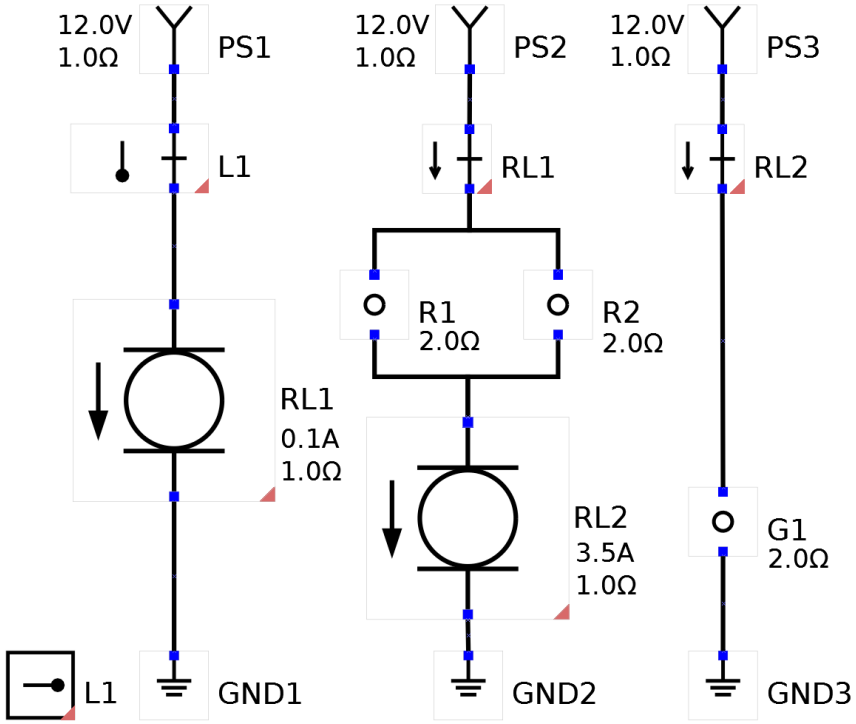


Figure 8.4: Principle schemata of the RIS R2G1 that controls the semaphore lights for a RIS level crossing. The RIS is formed by 4 sub-circuits not connected electrically — from left to right: a lever handle, the lever sub-circuit, the sub-circuit that controls the red lights of the traffic semaphore, and a sub-circuit that controls the green light of the train semaphore.

crossing (the red lights R_1 and R_2) and the semaphore for the train track (the green light G_1).

Each connected set of components in the RIS represents a sub-circuit (i.e., sub-circuits are not connected electrically to each other). In Figure 8.4 there are 4 sub-circuits — from left to right, the sub-circuits are the lever handle L_1 (note that the lever handle is by itself a sub-circuit), the sub-circuit that is controlled by L_1, the sub-circuit that controls the red lights, and the sub-circuit that controls the green light.

The sub-circuits are not connected electrically (i.e., with a wire), but are "connected" with some other means (e.g., mechanically, as for a lever, or magnetically, as for a relay coil). A component on one sub-circuit (e.g.,

a relay coil) opens or closes its contacts (e.g., the relay contacts) that are on other (electrically disconnected) sub-circuits. The principle schemata separates the representation of the components (e.g., a relay coil) and their contacts (e.g., the relay contact). In Figure 8.5 we show the symbols for a relay coil and its contacts. In a schemata, the components and their contacts are identified by name: the contacts for a relay coil named RL_1 will be also named RL_1. In a well formed schemata the same component name is used only for a component and its contacts (e.g., two relay coils cannot have the same name) and a contact must have a correspondent component (e.g., if a schemata has a contact named RL_1, it must also have a relay coil named RL_1). We say that there is a *logical connection* between a component and its contacts. The contact symbols in the diagrams further define

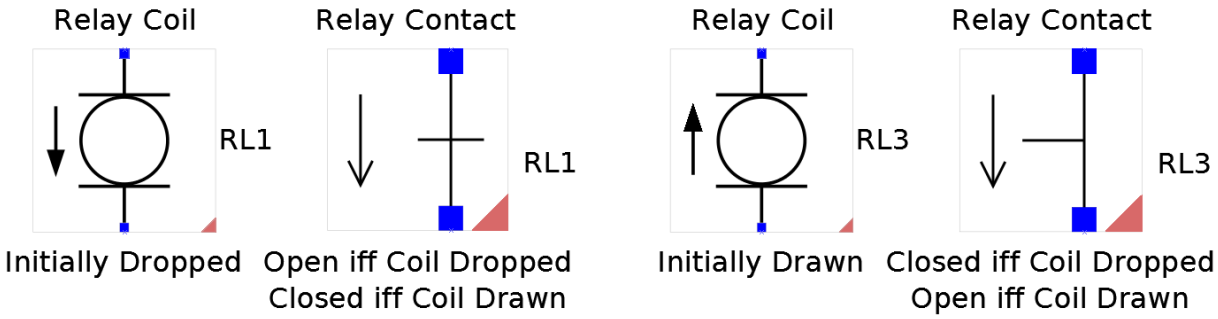


Figure 8.5: Symbols of the relay coils and their contacts.

when the contact should be open or closed. The two left-most components in Figure 8.5 are the relay coil RL_1 and an "open" contact RL_1 (in this case, the "open" qualifier identifies a contact that is open by default). The downward arrow shown on the left of the "open" relay contact specifies what will be the state of the contact (i.e. open or closed) depending on the state of its relay coil. In Figure 8.5, the contact RL_1 is open when the relay coil RL_1 is dropped and closed otherwise. Note that for the "closed" contact RL_3 of Figure 8.5 the downward arrow specifies that the contact is closed when the relay coil RL_3 is dropped, and open otherwise.

The graphical representation of the components further defines the electrical terminals of the components with blue square boxes and the electrical connections among terminals with black solid lines. The orientation of a component (important to determine the physical position, such as if a lever in the left, center, or right position), is uniquely represented with a red triangle in the bottom right corner of the component. The graphical representation describes also the initial state of switching components like lever handles and relay coils. For relay coils (see Figure 8.5) the initial state is determined by the upward or downward arrow at the left of the component, while for lever handles the initial state is the position (left, center, right) of the lever handle (e.g., in the schemata of Figure 8.4, the lever handle L_1 is initially in the left position).

In the RIS R2G1 we further have other electrical components like power generators (PS_1, PS_2, and PS_3) that generate a current on the sub-circuit and "ground components" (GND_1, GND_2, and GND_3) that determine the ground for each sub-circuit. The lever "open" contact L_1 in the RIS R2G1 is further closed only if the lever handle L_1 is in the center position (see the position of the lever on the left of the L_1 contact in Figure 8.4).

The RIS R2G1 implements a control logic that ensure that every time the green light is on (i.e. the train can travel through the track section with the level crossing), the red lights are also on (i.e. the cars have to stop at the level crossing). In the initial configuration of the RIS R2G1 both the red lights and the green lights are off. This is because the lever handle L_1 is in the left position, thus the lever contact L_1 is open, and hence no current flows in the sub-circuit and the coil RL_1 is dropped. Since the coil RL_1 is dropped, the contact RL_1 is open and no current flows through the red lights and the relay coil RL_2, which are respectively off and dropped. The contact RL_2 is further open and the green light is off. When an operator moves the lever handle L_1 to the center position she starts a sequence of

state changes in the RIS.

1. The operator moves the lever handle L_1 to the center position. This change instantaneously closes the lever contact L_1, and the current starts flowing on the coil RL_1.

2. After a small amount of time (the "transient" time of the relay), the relay coil RL_1 switches from the dropped to the drawn state, and the relay contact RL_1 closes. At this point, some current flows on the red lights and on the relay coil RL_2. The red lights turn on.

3. After a small amount of time, the relay coil RL_2 switches to the drawn state and the relay contact RL_2 closes, powering the green light that turns on.

8.2 Modeling approach

8.2.1 Choosing the modeling abstraction level for relays

The physical behavior of a RIS is determined by the complex electro-mechanical phenomena of the relays. The "stationary" relay's states are the drawn and dropped states. However, the real behavior of a relay is more complex due to inertial electro-mechanical phenomena: the transition between two stationary states is not instantaneous when the current on the relay's coil exceeds (or falls below) the threshold. Thus, we face the problem of modeling the relay's "transient states".

On the one hand a precise modeling of the "transient states" of the relays is challenging. First, such modeling requires complex differential equation; second, a RIS designer cannot reason precisely about the dynamic of the relay in the transient states. On the other hand, a purely "Boolean abstraction" approach that abstracts the physical quantities of the relay (e.g., the current on the coil) is also not adequate. Such abstraction does

not permit reasoning about the physical quantities and the relative time between events.

We adopt an intermediate approach where we model the physical quantities of the system but we abstract the "transient state" of the relays. We model that after the relay's current crosses the threshold the change of state of the relay happens in a non-deterministic (but bounded) time interval. This time interval is a known design parameter of a relay. Our approach preserves the actual stationary physics of the system and enables automatic reasoning on the relative time distance between events, that are two key aspects for the designer. In our ongoing project we identified this abstraction level as the suitable trade-off between the designer's needs and the availability of precise and efficient model checking algorithms.

8.2.2 Modeling Ris with Smdkn

Ris are networks of components electrically connected by means of the Kirchhoff conservation laws. For this reason, we model Ris with Smdkn. The main advantages of the Smdkn modeling are: (i) **Preserve the Ris structure.** We model the Ris network as a Smdkn that has the same network structure (i.e. electrical connections on the components' terminals). Thus, Ris designer can easily model the Ris principle schemata as a Smdkn. (ii) **Compositional modeling.** Smdkn allow us to define the component behaviors independently. Our modeling effort is thus limited to create a library of components for the Ris domain. (iii) **Smdkn are an expressive and flexible modeling language**. Smdkn allow us to model the behavior of switching components as hybrid automata. With hybrid automata we can easily model the "abstraction level" described above. (iv) **Availability of formal analysis techniques**. There already exist efficient formal verification techniques Smdkn [CMS16, CMS17] that we can apply off-the-shelf.

In the following, we describe in depth our modeling of the principle schemata as SMDKN, focusing on the components, their electrical connections, and the logical connections.

Components. We model a component in the RIS domain as a component in the SMDKN with a hybrid automaton. The hybrid automaton is standard [Hen96]: it defines a finite set of discrete modes and continuous variables. In each discrete mode the automaton defines with a differential equation how the contiguous variables change in function of time, and with a conjunction of Boolean inequalities the invariant conditions. Transitions between discrete modes model the instantaneous state changes. Both RIS and SMDKN components have electrical terminals. We follow the standard approach in *acausal modeling* [Fri14] to encode terminals with two variables, the flow and effort variables. In the electrical domain, the flow variable represents the current on the terminal, while the effort variable represent the potential on the terminal. Flow and effort variables will then be used to model the Kirchhoff conservation laws. The terminal implicitly has two continuous variables to represent flow and effort. Note that a component only exposes the effort and flow variables to the other components.

We describe in depth the modeling of a relay coil and of a faulty lamp. Both components are representative of the RIS library we developed that contains more than 100 components.

The model of the *delayed relay coil* shown in Figure 8.6 follows the abstraction level described above where the transient states of the relay coil are modeled non-deterministically. The two modes *Dropped* and *Drawn* of the automaton represent two stable states where the coil has completely actuated its contacts. The two modes *Drawing* and *Dropping* encodes the transient states of the coil. The automaton uses a clock variable *clock* to encode the bounded and non-deterministic transition delays between the

stable modes. In particular, the automaton transition from the *Dropped* to the *Drawn* mode only fires when the electrical current I through the coil continuously exceeds the current threshold I_{th} for a non-deterministic time within the specified time interval $[\Delta T_-, \Delta T_+]$. The same happens for the transition from the *Drawn* to the *Dropped* mode.

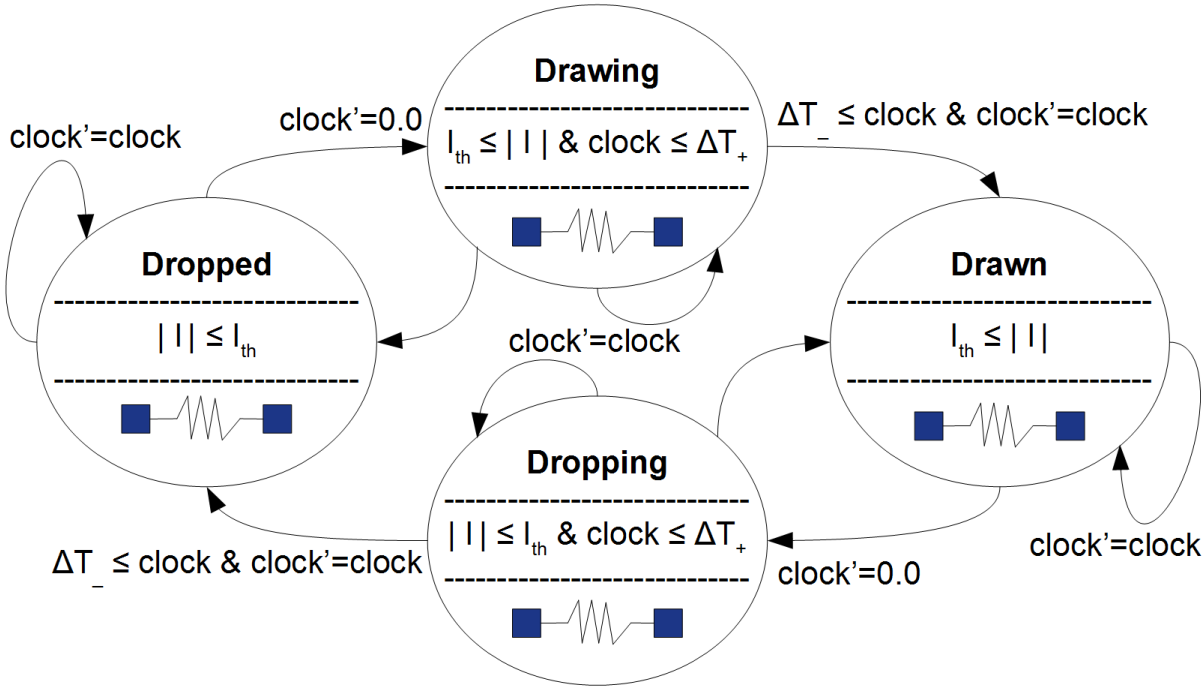


Figure 8.6: Hybrid automaton of the delayed relay coil.

Figure 8.7 shows the model of a *faulty lamp*, a lamp that can fail either creating a short-circuit or opening the circuit. The *Nominal* mode encodes the correct behavior of the lamp, which behaves as an ohmic load resistor. The automaton encodes the two fault conditions in the *FaultShort* and *FaultBlown* modes, where the lamp behaves respectively as a short-circuit and as an open circuit. The automaton can non-deterministically transition from the nominal mode to the two faulty modes. Since the lamp does not exhibit commutation delays, the hybrid automaton does not have continuous variables.

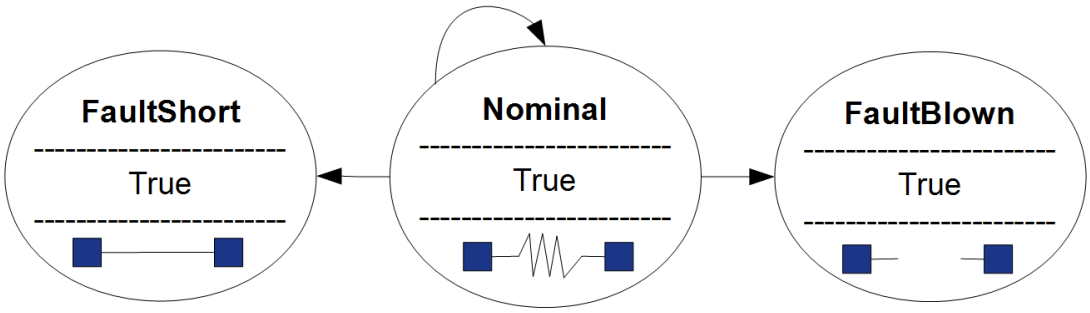


Figure 8.7: Hybrid automaton of the faulty-lamp.

Physical connections. The semantics of the terminal connections follows the Kirchhoff's conservation laws. Given a set of connected terminals, all the effort variables of the terminals take the same value, and the sum of all the flow variables of the terminals equals zero. The SMDKN semantics already considers the Kirchhoff's law.

Logical connections. We model the logical connection among two components (e.g., a relay coil and one of its contacts) with additional synchronization constraints among the discrete modes of the hybrid automata of two components. For instance, for the relay coil RL_1 and the relay contact RL_1 of Figure 8.5 the constraint encodes that the coil is in the *Dropped* mode if and only if the contact is in the *Open* mode, and in the *Closed* mode otherwise. Similarly, for the lever handle L_1 and its lever contact L_1 of Figure 8.4, we say that the handle is in the *Center* mode if and only if the contact is *Closed* mode.

Physical behavior of the running example. We present the relevant electrical behavior of the R2G1 system when lamps can fail either blown or short-circuited. The relay coil RL_2 of Figure 8.4 senses the electrical current I_{PS_2} flowing through the parallel connection of the red lamps R_1 and R_2 in order to monitor their status. The current threshold of the coil RL_2 should be properly set to prevent inadvertent activation of the green lamp

G_1 when the red lamps are either off or faulty. Table 8.1 shows the value of the current I_{PS_2} as a function of the 9 possible system modes resulting from the cross product of the 3 modes of the red lamps (see Figure 8.7).

System mode	**Current I_{PS_2}**
Both red lamps failed blown	0.0 Ampere
One red lamp failed blown, one red lamp nominal	3.0 Ampere
Both lamps nominal	4.0 Ampere
At least one red lamp failed short-circuited	6.0 Ampere

Table 8.1: Values of the electrical current I_{PS_2} sensed by the relay coil RL_2 when the red lamps are power supplied by the closed relay contact RL_1.

To detect the simultaneous activation of the red lamps, the current threshold of the relay RL_2 must be set in the interval $]3.0A, 4.0A[$, for instance to $3.5A$. Notice that, in the system design of Figure 8.4, the configurations "both lamps nominal" and "at least one red lamp failed short-circuited" are indistinguishable to the coil RL_2 because in both cases the current I_{PS_2} exceeds the coil threshold of $3.5A$. In the following section, we discuss the implication of this consideration on the overall system safety and we show how the proposed methodology supports the designer on this kind of quantitative reasoning.

8.3 Formal analysis

In a RIS, the agents determine their next action observing the state of the field devices. Thus, the agents observe a partial-state of the system because the internal state of the control logic is hidden from their point of view. Nevertheless, the correctness of the signaling protocol is implicitly dependent from the implementation of the relay logic.

In our methodology, we propose to analyze the system at two levels of detail: at the higher *railway level* we consider only high-level properties

over the field devices (e.g., the lamp emits light, the barrier is closed), despite the technological details of the control logic; at the lower *physical level* we consider properties that investigate the internal technological aspects of the control logic and of its physics (e.g., two terminals must be short-circuited when a relay is in a specific mode). This layered approach reduces the total effort to specify properties: the properties at the railway level are independent from the implementation of the control logic and can be reused for multiple control logic implementations.

8.3.1 Properties specification

a property at the physical level predicates on low level aspects of the system such as physical quantities and operating modes of the components. Focusing on the electrical domain, we can predicate either on the voltage drop ΔV across a pair of terminals, or on the current I that flows through a terminal. A similar approach holds in the mechanical domain replacing *current* and *voltage* with *torque* and *angular velocity*. A property can further predicate on the operational modes of the components.

A railway property is automatically mapped onto a combination of physical properties, hiding its implementation details. For instance, consider the sentence *"the lamp* G_1 *emits light"*. Since a lamp is electrically equivalent to an ohmic load resistor, the property is equivalent to *"the lamp* G_1 *consumes electrical power"* that in turns is equivalent to the first-order logical formula $I_{\mathrm{G}_1} \neq 0.0 \wedge \Delta V_{\mathrm{G}_1} \neq 0.0$. Notice that in the context of physical reasoning it is necessary to predicate on *both* currents and voltage drops in order to distinguish the nominal behavior of the lamp from the faulty ones (i.e. those in which the lamp is power supplied, but does not emit light). In fact, a short-circuited lamp is traversed by a non-null current ($I_{\mathrm{G}_1} \neq 0.0$), but its voltage drop is zero ($\Delta V_{\mathrm{G}_1} = 0.0$); similarly, a blown lamp is traversed by a null current ($I_{\mathrm{G}_1} = 0.0$) even if its voltage

drop is different from zero ($\Delta V_{\mathrm{G}_1} \neq 0.0$). In our specification settings, we could also refine the property exploiting detailed information available to the designer. Assuming to know the range of nominal currents absorbed by the lamp (e.g., from its data sheet), we could rewrite the predicate $I_{\mathrm{G}_1} \neq 0.0$ into a more precise one such as $1.5 \leq |I_{\mathrm{G}_1}| \leq 2.3$.

8.3.2 Analysis of the running example

in the following we demonstrate the need of the quantitative reasoning, which is enabled by our modeling approach, using the RIS R2G1 of Figure 8.4. We further consider variants of the R2G1 model changing the fault model for the red lamps and the current threshold of the relay coil RL_2. The red lamps may either not fail, or the red lamps may blown (see the *FaultBlown* state in Figure 8.7), or the red lamps can introduce a short circuit (see the *FaultShort* state in Figure 8.7). The current threshold on the relay coil RL_2 may be either 2.5A, or 3.5A, or 4.5A. We consider the reachability property $RP :=$ *"the green lamp* G_1 *can emit light"*, and the safety property $SP :=$ *"if the green lamp* G_1 *emits light, then both red lamps* R_1 *and* R_2 *emit light"*. We expect RP to hold for R2G1, witnessing an execution scenario where green lamp is on, and SP to hold to ensure the safety of the R2G1 system. The verification results are available in Table 8.2.

When the current threshold of the relay coil RL_2 is over-dimensioned to 4.5A, the unexpected verification of the property RP proves that the green lamp cannot emit light because the relay contact RL_2 will never supply power to the lamp (rows 3, 6). Decreasing the threshold, RP always holds and this fact guarantees that the green lamp can turn on.

When the current threshold is under-dimensioned to 2.5A, the safety property SP is violated in the system variant with blown lamps (row 4). The counterexamples returned by the model checker provide execution

Nr.	R2G1 Variants		Verification results	
	Faults	**RL_2 thresh.**	**RP**	**SP**
1	None	2.5A	Hold	Hold
2	None	3.5A	Hold	Hold
3	None	4.5A	Doesn't hold	Hold
4	Blown	2.5A	Hold	Doesn't hold
5	Blown	3.5A	Hold	Hold
6	Blown	4.5A	Doesn't Hold	Hold
7	Short	2.5A	Hold	Doesn't hold
8	Short	3.5A	Hold	Doesn't hold
9	Short	4.5A	Hold	Doesn't hold

Table 8.2: Verification results (property holds or does not hold) on variants of R2G1 introducing faults on the red lamps and changing the current threshold on the relay coil RL_2.

scenarios able to reach the violation, but do not represent an exhaustive analysis. To determine all the minimal configurations of faults that lead to the violation, we perform formal safety assessment to compute fault-trees. For the system variant of row 4, the fault-tree of the safety property SP shows two possible fault configurations: when one red lamp fails blown, the other red lamp can still emit light absorbing 3.0A (see Table 8.1) from the power supply PS_2. The 3.0A current exceeds the under-dimensioned threshold of 2.5A, thus the relay RL_2 inadvertently supplies power to the green lamp, violating the safety property. We fix this design flaw setting the coil threshold to 3.5A (row 5).

Unfortunately, the safety violation still occurs when the lamps fail short-circuited (row 8). The safety assessment process reveals that if any red lamp fails short-circuited, a current of 6.0A is drawn from PS_2 (see Table 8.1), and the relay coil RL_2 is again deceived. This design flaw cannot be fixed by simply adjusting the electrical parameters of the system, but requires the upgrade of the entire design as shown in Figure 8.8. In the

system upgrade, the additional relay coil RL_3 is *Drawn* when the current I_{PS_2} exceeds the threshold of 4.5A, that makes its contact RL_3 open, thus preventing the green lamp from turning on if a red lamp is short-circuited.

8.3.3 Need of quantitative modeling for verification

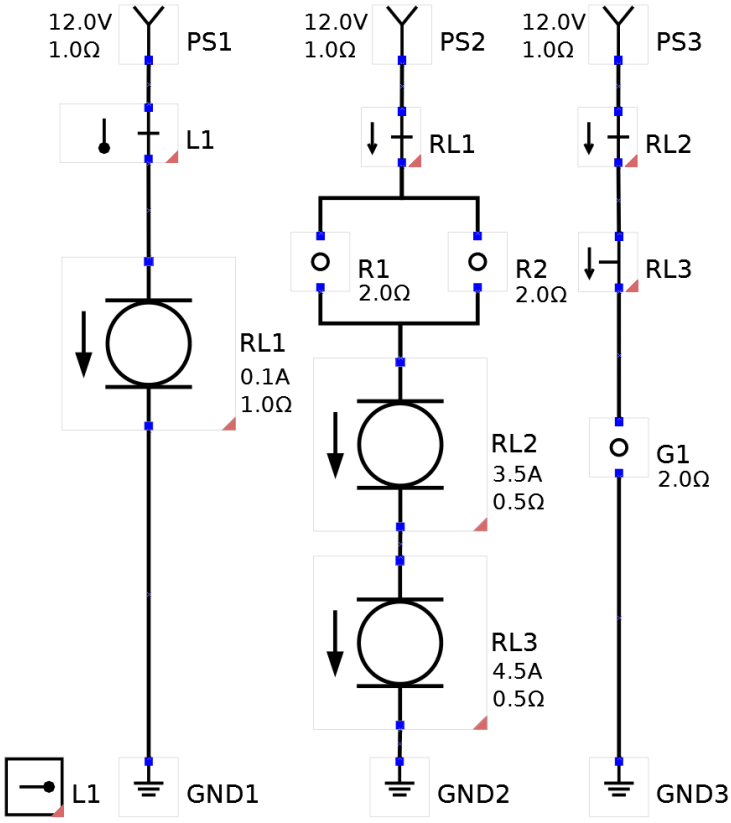


Figure 8.8: Upgraded design of the Ris R2G1 from Figure 8.4

We make a small digression to report the main limitations we encountered while applying the traditional Boolean modeling approach (i.e. the one based on the concept of conductive paths) that led us to this work. Referring to the upgraded R2G1 design of Figure 8.8, Figure 8.9 shows the value of the current I_{G_1} flowing through the green lamp G_1 as a function of the current I_{PS_2} sensed by the relay coils RL_2 and RL_3. Our physical modeling approach (Figure 8.9-(2)) is able to properly discriminate the

faulty scenarios (i.e. $I_{\mathrm{PS}_2} < 3.5A$ and $I_{\mathrm{PS}_2} > 4.5A$, where $3.5A$ is the RL_2 threshold and $4.5A$ is the RL_3 threshold), keeping the green lamp properly turned-off (i.e. $I_{\mathrm{G}_1} = 0.0A$). Differently, the expressiveness of the Boolean approach ((Figure 8.9-(1))) cannot discern between different values that are greater than zero. This means that, for every current $I_{\mathrm{PS}_2} > 0.0A$, the relay coils RL_2 and RL_3 would be considered always Drawn, resulting in a spurious behavior with the green lamp always turned-off.

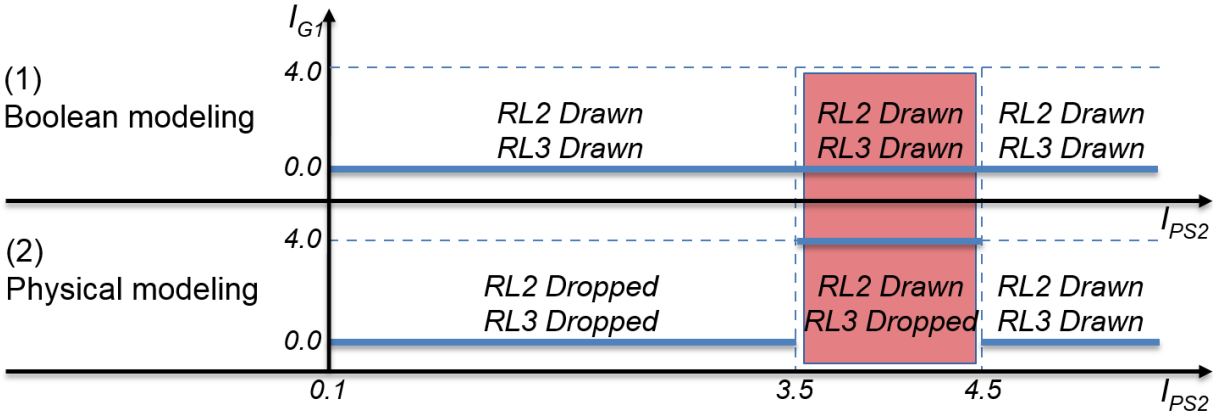


Figure 8.9: Spurious behavior on the green lamp G_1 introduced by the Boolean modeling. The relay coils RL_2 and RL_3 are permanently Drawn, and keep G_1 always turned off.

8.4 Tool Chain

The proposed methodology was implemented in a tool chain composed of various blocks and represented in Figure 8.10.

The first block is a graphical front end (Figure 8.11) based on a customization of the DIA [GNO17] modeling environment. The palette of the front end supports over 100 distinct graphical symbols, corresponding to a subset of the components that can be found in RIS according to the Italian regulation. Each symbol is associated to an internal data structure, where parameters of various kinds are associated (e.g. delay in response time, resistance, and angular velocity).

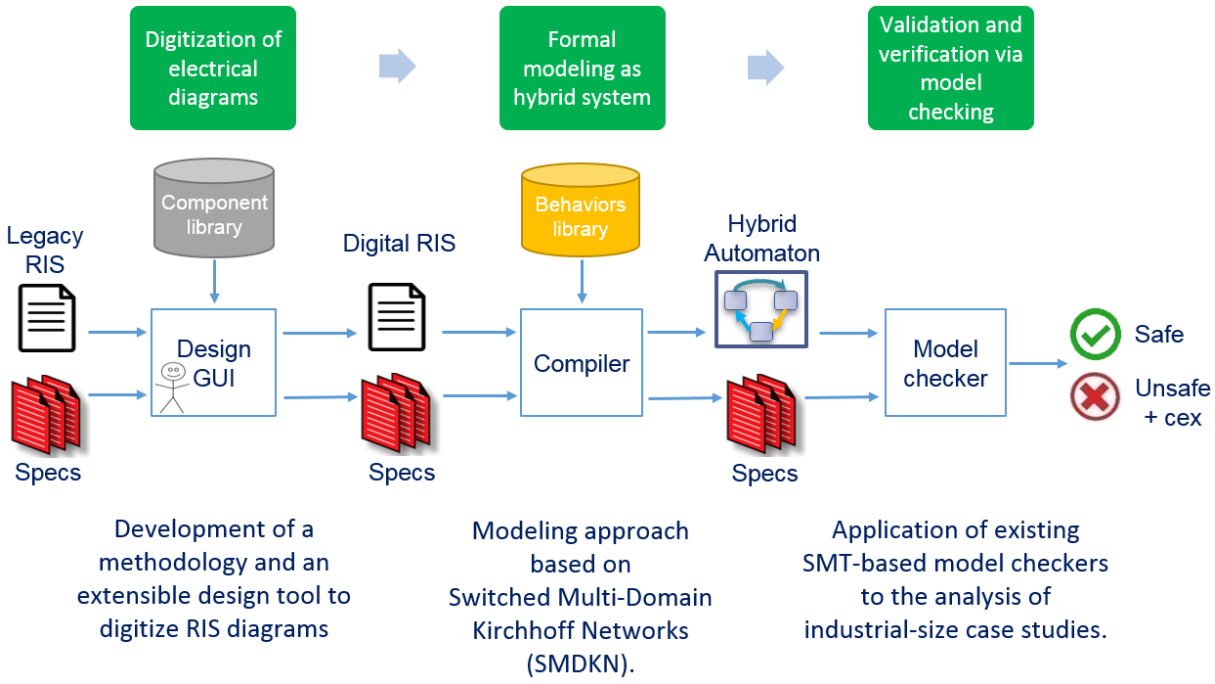


Figure 8.10: Overview of the tool chain implemented for the analysis of RIS.

The front end supports the connection between components, and carries out a number of sanity checks to pinpoint errors such as dangling terminals, missed components in logical connections, and conflicting logical connections between incompatible symbols. The front end also supports the definition of railway predicates representing some relevant physical conditions. Properties are expressed in form of linear temporal logic over both railway and physical predicates.

The second block is a compiler from SMDKN to hybrid automata network, symbolically expressed in the HYDI language [CMT11b]. The compiler is written in Python, and implements the conversion traversing the network based on an extensible library of behavioral component descriptions.

The third block is the HYCOMP model checker [CGMT15], that processes the resulting HYDI network and carries out the required analyses, leveraging various SMT-based engines for model checking [CCD+14], to-

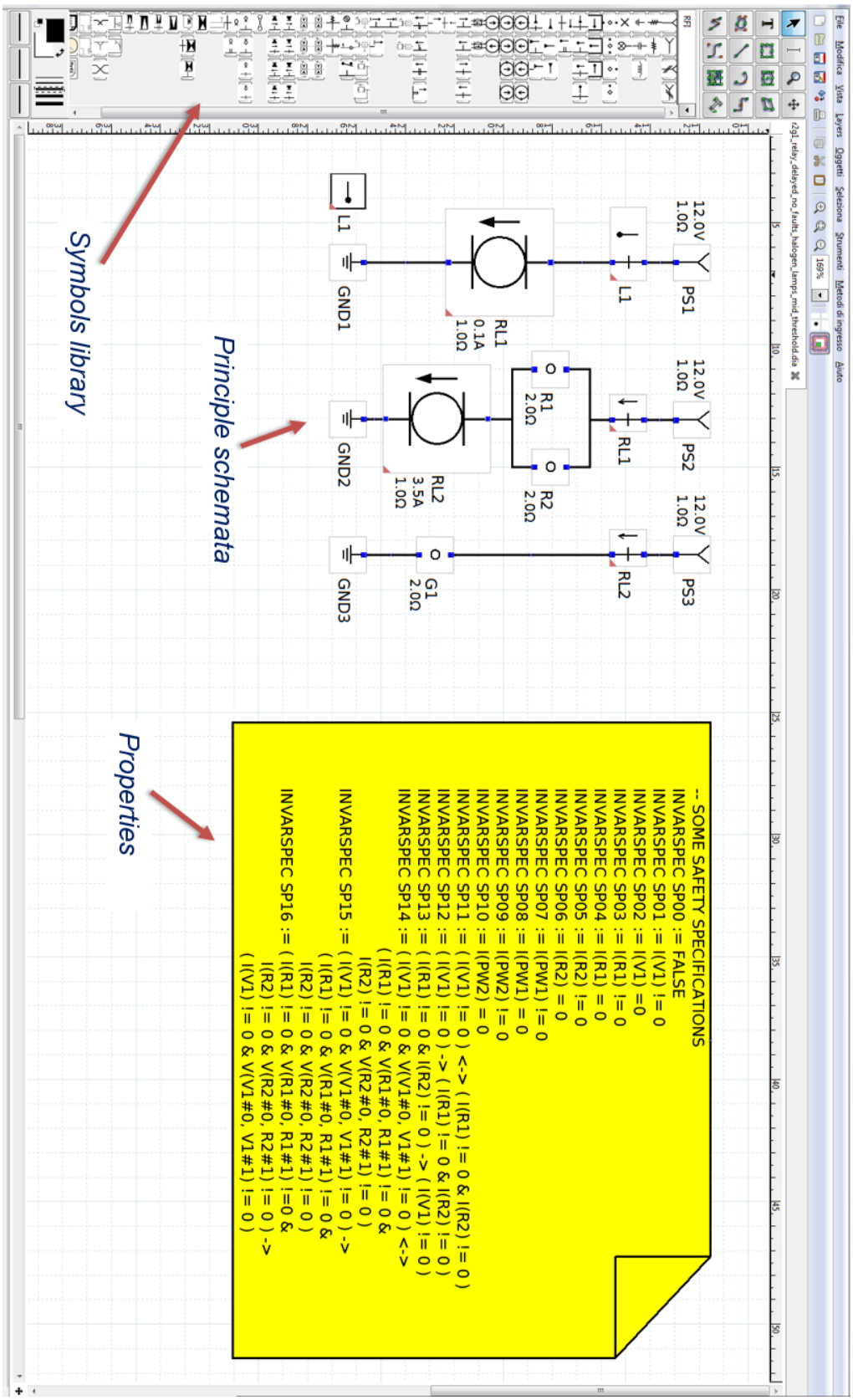


Figure 8.11: Front end of our design tool.

gether with xSAP [BBC+16] for safety analysis and fault-trees production.

8.5 Experimental Evaluation

8.5.1 Benchmarks

We evaluated the proposed methodology analyzing a scalable, industrial-size RIS referred to as RISCS. Figure 8.12 shows a simplified layout of the RISCS, omitting both the electrical connections among devices and other confidential details of the relay logic. The $\text{RISCS}_{[i]}$ system represents a railway section along a bidirectional train line containing a sequence of i level crossings, with $1 \leq i \leq 10$. The section is protected on each track side by a warning and a protection semaphore. The warning/protection semaphores have three yellow/red lamps (WYL/PRL) and two green/green lamps (WGL/PGL). The lamps of the same color are electrically connected in parallel to improve the redundancy of each semaphore. Every level crossing is protected on each street side by a barrier (LCB) and by a vehicular semaphore consisting of one red lamp (LCL). The presence of the train along the line is detected by means of the train approaching pedals (TAP) and of the train detection pedals (TDP). The maintainers can completely/partially disable the section acting on several maintenance levers (GML, TAML, LCML) at the maintenance place. The train dispatcher can activate the section acting on the section enabling lever (SEL) at the train station. The relay logic is electrically connected to all the devices shown in Figure 8.12. The relays sense the electrical currents flowing through every connected device and actuate a specific control sequence, transferring energy between the devices. For instance, when the train pushes the left train approaching pedal (left TAP), closing its sub-circuit, the logic checks the magnitude of the current flowing through the level crossing lamps (up/-down LCL) of the vehicular semaphores, and, if all the lamps work properly,

Figure 8.12: Physical layout of the RISCS$_{[i]}$ case study. **Legend:** Warning Yellow Lamp (WYL), Warning Green Lamp (WGL), Protection Red Lamp (PRL), Protection Green Lamp (PGL), Level Crossing Lamp (LCL), Level Crossing Barrier (LCB), Train Approaching Pedal (TAP), Train Detection Pedal (TDP), Level Crossing Maintenance Lever (LCML), Train Approaching Maintenance Lever (TAML), General Maintenance Lever (GML), Section Enabling Lever (SEL).

the logic powers on the engines of the barriers (LCB) to start the lowering sequence.

We modeled the RISCS case studies with our tool, selecting and modeling the components and their parameters, their interconnections, and verifying properties of interest. The overall modeling task lasted for about 3 weeks, including the creation of a reusable behavioral component library.

The largest system $\text{RISCS}_{[10]}$ contains 141 power supplies, 22 resistors, 113 relays, 15 levers, 12 pedals, 678 contacts, 40 lamps, 23 maintenance lights, and 54 circuit breakers (printed on twenty A4-sheets of paper). These components are distributed over 125 sub-circuits. The conversion of the corresponding SMDKN into hybrid automaton returns an SMT encoding that uses 437 Boolean variables to encode the discrete part, and 6281 real-valued variables to encodes the physical part. Clearly, the size of the state-space makes traditional manual inspection extremely time-consuming, expensive, and unfeasible in practice.

We present the results of the analysis on the nominal and faulty variants of the RISCS system, where up to 80 electrical faults (i.e. blown or short-circuited lamp) are injected on the 40 semaphore lamps in the case of the $\text{RISCS}_{[10]}$ benchmark.

8.5.2 Verification

We model checked the RISCS system against 190 invariant properties, running the two verification algorithms IC3 [CGMT16] and BMC [BCC+03] that represent complementary techniques to either verify or falsify properties. We run the experiments on a 3.5 GHz cpu with 16GB RAM, with time out (TO) set to 3600 seconds. About half of the properties represent scenarios that are supposedly feasible, and are used to validate the system design. The first validation round reported that some scenarios were found to be (unexpectedly) unfeasible. Upon fixing some buggy components in

the behavior library, all the scenarios were proved to be feasible, within the timeout of 3600s, in both the nominal and faulty case. The resulting execution traces were analyzed and validated by the domain experts. Examples of scenario include that every lamp of every semaphore can be turned on and then off (Figure 8.13), or that every barrier can be completely lowered and then raised (Figure 8.14).

Figure 8.13: Graphical representation of the property "Every semaphore lamp can turn off/on".

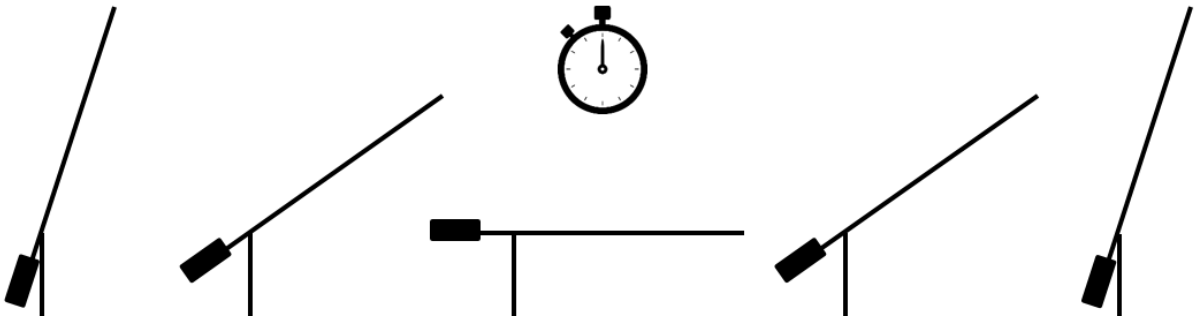


Figure 8.14: Graphical representation of the property "A complete lowering-raising sequence of the barriers is possible".

The remaining properties express the absence of safety violations. Most of them are verified in the nominal case within the timeout, except for three properties on the synchronization among the warning and protection semaphores.

Some relevant properties expressing the proper synchronization between the semaphore lights and the barriers positions hold also under the non-nominal case (i.e. when components are subject to faults). For instance, the model guarantees that the green lamps of the protection semaphores are off when the level crossing barriers are not completely closed (Figure 8.15).

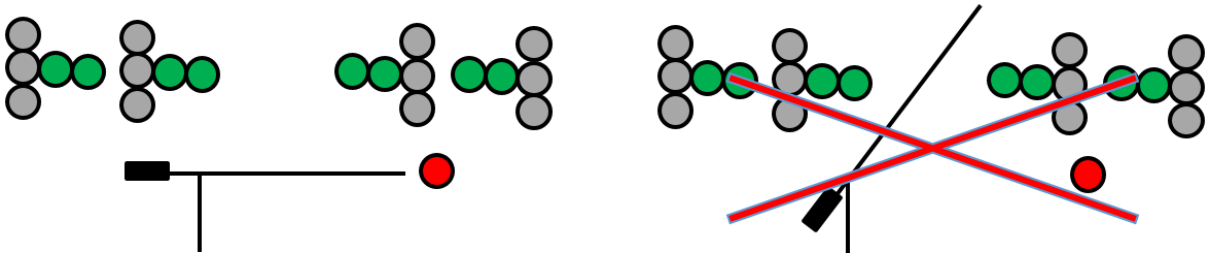


Figure 8.15: Graphical representation of the property "The train semaphores never grant access to the train when the barrier is not completely closed".

Moreover, we are guaranteed that the colors of every semaphore are turned on in a mutually-exclusive way (Figure 8.15).

Figure 8.16: Graphical representation of the property "The two semaphore colors are always mutually exclusive".

Noteworthy, we successfully verified an electrical safety requirement (a low-level electrical property) prescribed by the national regulation: the level crossing lamps are short-circuited when the barriers are open and resting to prevent inadvertent activation.

59 safety properties were violated in the faulty case. Some of them check for each semaphore if there is always at least one lamp turned on. Of course, in case of multiple lamp faults, this condition cannot be avoided because all the lamp might fail. With formal safety analysis, we compute the fault tree responsible for the violations. For a warning semaphore, the fault tree (Figure 8.17) shows that the violation might be reached in 7 distinct circumstances: either all yellow lamps are blown, or all green lamps are blown, or at least one yellow lamp is short-circuited, or at least one green lamp is short-circuited. The first two circumstances represent fault configurations of size 3 and 2, respectively the number of yellow and

green lamps, that would be hard to spot by manual inspection.

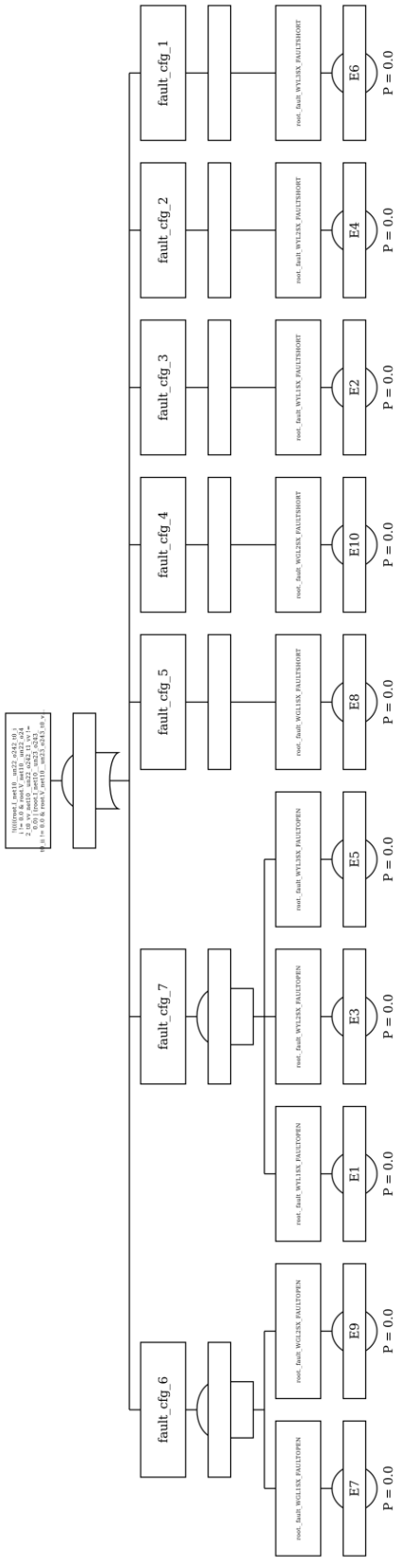


Figure 8.17: Fault-tree automatically compute by formal safety analysis tool for the violation of the safety property "At least one semaphore lamp is always on" in case of multiple lamp faults From left to right, the fault-tree shows one minimal cutset of size 2, one minimal cutset of size 3, and 5 minimal cutsets of size 1.

Chapter 9

Related Works

Multi-Domain Kirchhoff Networks are widely used in various engineering applications [YONS16, EOS97, DMBF13]. Different tools support the acausal modeling phase [Fri14, Mat18, Bro99], also for networks with discrete switches. The main analysis tools are based on numerical simulation and use numerical integration. Although simulation provides high scalability and enables the analysis of complex dynamics [Ria08, Ben14, Ska01], a preliminary validation of the network modes is not provided. Therefore, a hidden inconsistent mode can be discovered *only* if the user designs a simulation trace that is able to reach it. Furthermore, numerical simulators (e.g. [Mat18]) restrict the use of components equipped with ideal behaviors, leading to the model pollution due to parasitic effects, that are hard to quantify and deviate the simulation results from the intended nominal behavior. The INTO-CPS project [LFW+16] realized an integrated tool chain for comprehensive model-based design of cyber-physical systems. The tool chain supports the multidisciplinary, collaborative modeling of CPSs from requirements, through design, down to source code generation and deployment on hardware. Model exchange and co-simulation of dynamic model is supported by a tool-independent standard named Functional Mock-up Interface (FMI) [BOA+11]. Although the INTO-CPS tool chain mainly

focuses on co-simulation, some model checking features are provided as well. The continuous dynamics of the cyber-physical system is abstracted using discrete-event abstractions which are expressed as state machines. The overall cyber-physical system is then represented as the composition of discrete-event components only. The tool chain provides SMT-based reasoning on this network of components. INTO-CPS supports three different kinds of abstractions of continuous dynamics: based on intervals, based on the derivative of the signal values, and based on interval abstractions specifically generated from concrete simulations. Our SMT-base approach is positioned at a higher level with respect to the abstraction of a specific continuous dynamics performed by the model checker. Our validation and reformulation approach aims at reformulating the DAE dynamics into a hybrid automaton wit ODE amenable to formal verification with the existing SMT model checkers. Our reformulation process is precise in the sense that the abstraction of the continuous dynamics is not involved. Additionally, our approach is general in the sense that does not rely on any particular model checking engines and abstraction technique. We think that our approach might be complementary with respect to the co-simulation approach.

In the following, we focus on works based on formal methods.

The closest related work is [CMS16], that presents a method to convert Switched *Electrical* Kirchhoff Networks (SEKN) into hybrid automata. The work proposed here is more general than [CMS16] in two respects. First, we are able to deal with *multi-domain* networks, enabling mechanical, electrical and hydraulic domains, and their combination, whilst [CMS16] is restricted to electrical networks. Second, the method in [CMS16] is only able to produce a hybrid automaton if the electrical network fulfills the conditions of existence and determinism in *all the modes* and for *all the variables*, while here we analyze SMDKN

with *non-deterministic* algebraic variables as well. Both extensions are made possible by the adoption of a theoretical settings that is significantly more general than the domain-specific topological approach on the network graph used in [CMS16]. We remark that all the experiments presented in the previous chapter are based on benchmarks that are out of reach for the method in [CMS16]. In [BBGJ15], a framework for generating hybrid automata benchmarks from a hydraulic domain is presented. This work is only seemingly related to ours. The domain knowledge in [BBGJ15] (e.g. that a pump cannot draw a constant flow from an empty tank) appears to be hard-coded in the generation scripts; in our case, the detection of these conditions and the generation of the hybrid automata are direct consequence of the algebraic approach applied to the network description. As discussed in the experimental evaluation, our approach is able to deal with a significantly larger class of benchmarks than those in [BBGJ15], and also to automatically identify invalid modes in the network, reasoning on its algebraic properties.

Most of the formal verification tools are unable to deal with DAE. An exception is KeymaeraX [FMQ+15], a theorem prover for hybrid systems represented with Differential-Algebraic Equations. In principle, the KeymaeraX proof system can support the proof of safety properties over Smdkn, by means of compositional reasoning. Key differences with our approach are that KeymaeraX is not fully automatic, and has no specific methods to address the validation problem.

The existing tools for formal verification of hybrid systems [Alu11] do not directly consider Multi-Domain Kirchhoff Network, but work on hybrid automata [Hen96]. Tools like *SpaceEx* [FGD+11] or *Flow** [CÁS13] work on an *explicit* representation of the system and hence they suffer from the explosion in the number of modes of the system. Other tools [Tiw12, CGMT15, KGCC15, BKG15] reason on the symbolic rep-

resentation of the system. HybridSAL [Tiw12] and HyComp[CGMT15] analyze linear hybrid systems whose continuous dynamics is specified with a linear ODE. dReach [KGCC15, BKG15] can be used to either perform Bounded Model Checking or apply induction to verify a system expressed with ODEs. From a DAE-based network, our reformulation step produces this kind of formal models.

Other verification techniques focus on analog-mixed-signals circuits [ZTB06, DDM04, FKRM06, LAH+15, ZSS12]. They take the hybrid automata representation of the electrical circuit, so do not face the validation and reformulation problems. Additionally, they do not consider multi-domain networks and perform an analysis explicit in the modes that might exponentially blow-up.

Other approaches exist to generate a formal representation from Simulink and other causal component-based modeling languages [MMBC11, MF16]. This causal semantics considers systems represented as a connection of input-output functional blocks, posing a major obstacle to the modeling of Smdkn. Our work differs from those approaches since we natively accept the more suitable *acausal* component-based modeling, that, on the other side, requires to tackle the reformulation problem.

From the point of view of the railway case study, formal methods have been heavily applied in the railway domain. Important works on the verification of interlocking systems include (but are not limited to) [HØ16, HHP17, FHM17, HCC+00, CCL+12, CGM+98]. These works are not related, since they do not consider the specific case of relay circuits.

To the best of our knowledge, no works address the verification problem of a Ris based on its hybrid physical behavior, except our work [CCM+18]. Closely related works are [BFBT16, BFB+13, HKB11, Eri04]. While we model the evolution of continuous signals over time, the above works model

Boolean signals evolving over discrete time. Furthermore, these works assume that the interaction with the environment is limited to one input per cycle to ensure that the internal micro-sequence of relay commutations started from an input command is fully extinguished (run to completion) before the arrival of the next input. In [Eri04], two interesting observations are made. First, the discrete model of time does not support reasoning about relative time distances (e.g., between events, and on parasitic delays); second, the restriction on the number of inputs per execution cycle only works under the assumption that the control logic reacts "quickly enough" to every change in its environment. Our approach overcomes both limitations adopting a continuous model of time and not imposing restrictions on the environment. Thus, we deal with an arbitrary number of concurrent inputs and analyze the effect of inputs received in the middle of an internal micro-sequence.

We now analyze these works in more detail. The works [BFBT16, BFB+13] present a practical approach to the Ris safety certification. A *Boolean* model is extracted from the Ris and analyzed via SAT-based abstraction-refinement. Our SMT-based approach enables more fine grained analyses, modeling the precise physics of the system and preventing spurious behaviors introduced by the Boolean abstraction. The work [HKB11] builds a Boolean model based on the abstraction concept of *conductive path*: a relay coil is drawn iff all the conduction conditions along a conductive path from a power supply to the coil are satisfied. This approach is subject to several limitations: it is only valid under some assumptions on the system physics (e.g., all the power supplies are always up and running); it requires the enumeration of a potentially exponential number of conductive paths; it does not permit a quantitative reasoning (e.g., how much current flows through a conductive path). There is only one work [Eri04] that considers risk analysis and the effects of single-mode

faults on the system safety. These faults are Boolean and limited to the discrete state of relays (e.g., stuck at dropped/drawn). In our work we allow the designer to specify a larger class of faults, both on the discrete and physical state of components, with no limitation on the contemporaneity of fault occurrences.

Chapter 10

Conclusion

We presented an SMT-based method for the formal analysis of Switching Multi-Domain Kirchhoff Networks (SMDKN), that is able to automatically validate and reformulate a SMDKN into a symbolic Hybrid Automaton, amenable to be formally verified with the existing model checkers. The approach covers networks spanning multiple physical domains and exhibiting non-deterministic behaviors, achieving substantial improvements over a pure SMT-based approach by leveraging general results in linear algebra. We implemented and evaluated the SMT-based procedures to validate and reformulate the network, demonstrating the potential of complete verification workflow on real-world systems and on a railway case study developed in collaboration with the Italian train company.

We plan to extend the approach to incorporate networks with discontinuous state variables [Mea97], produce a network of HA instead of a monolithic HA.

In the railway case-study we experimented an approach based on this work to understand legacy relay circuits in the railway domain. We rely on an accurate representation at the physical level in form of Switched Kirchhoff Networks, that is then reduced to a symbolically represented network of hybrid automata, and then analyzed by means of SMT-based

model checking. The experimental evaluation demonstrates the precision and scalability of the analyses. The proposed methodology is at the core of an ongoing research project aiming at the in-the-large analysis of legacy railway interlocking and the open specification of computer-based solutions. Directions for future research include the definition of a library of property patterns, the definition of specific verification engines, and the integrated animation of counterexamples.

Bibliography

[Alu11] Rajeev Alur. Formal verification of hybrid systems. In *EMSOFT 2011*, 2011.

[Axl97] Sheldon Jay Axler. *Linear Algebra Done Right*. Undergraduate Texts in Mathematics. Springer, 1997.

[BBC+16] Benjamin Bittner, Marco Bozzano, Roberto Cavada, Alessandro Cimatti, Marco Gario, Alberto Griggio, Cristian Mattarei, Andrea Micheli, and Gianni Zampedri. The xSAP safety analysis platform. In *Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2016, Proceedings*, pages 533–539, 2016.

[BBGJ15] Stanley Bak, Sergiy Bogomolov, Marius Greitschus, and Taylor T. Johnson. Benchmark generator for stratified controllers of tank networks. In *ARCH*, 2015.

[BCC+03] Armin Biere, Alessandro Cimatti, Edmund M. Clarke, Ofer Strichman, and Yunshan Zhu. Bounded model checking. *Advances in Computers*, 58:117–148, 2003.

[Bea15] Marco Bozzano and et al. Formal design and safety analysis of AIR6110 wheel brake system. In *CAV*, 2015.

[Ben14] Peter Benner. *Large-scale networks in engineering and life sciences*. Springer, Birkhäuser Mathematics, 2014.

[BFB+13] Andrea Bonacchi, Alessandro Fantechi, Stefano Bacherini, Matteo Tempestini, and Leonardo Cipriani. Validation of railway interlocking systems by formal verification, A case study. In *Software Engineering and Formal Methods - SEFM 2013*, pages 237–252, 2013.

[BFBT16] Andrea Bonacchi, Alessandro Fantechi, Stefano Bacherini, and Matteo Tempestini. Validation process for railway interlocking systems. *Sci. Comput. Program.*, 128:2–21, 2016.

[BKG15] Kyungmin Bae, Soonho Kong, and Sicun Gao. SMT encoding of hybrid systems in dReal. In *ARCH14-15*. EasyChair, 2015.

[BOA+11] Torsten Blochwitz, Martin Otter, Martin Arnold, Constanze Bausch, Christoph Clau, Hilding Elmqvist, Andreas Junghanns, Jakob Mauss, Manuel Monteiro, Thomas Neidhold, Dietmar Neumerkel, Hans Olsson, Jrg-Volker Peetz, and Susann Wolf. The functional mockup interface for tool independent exchange of simulation models. pages 105–114, 03 2011.

[Bro99] Jan F. Broenink. 20-sim software for hierarchical bond-graph/block-diagram models. *Simul. Pr. Theory*, 7(5-6):481–492, 1999.

[BSST09] Clark W. Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. Satisfiability modulo theories. In *Handbook of Satisfiability*. IOS Press, 2009.

[BW14] Frédéric Boniol and Virginie Wiels. The landing gear system case study. In *ABZ 2014: The Landing Gear Case Study*, pages 1–18. Springer International Publishing, 2014.

[CÁS13] Xin Chen, Erika Ábrahám, and Sriram Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *CAV*. Springer, 2013.

[CCD+14] R. Cavada, A. Cimatti, M. Dorigatti, A. Griggio, A. Mariotti, A. Micheli, S. Mover, M. Roveri, and S. Tonetta. The nuXmv symbolic model checker. In *CAV*, volume 8559 of *Lecture Notes in Computer Science*, pages 334–342, 2014.

[CCL+12] Alessandro Cimatti, Raffaele Corvino, Armando Lazzaro, Iman Narasamdya, Tiziana Rizzo, Marco Roveri, Angela Sanseviero, and Andrei Tchaltsev. Formal verification and validation of ERTMS industrial railway train spacing system. In *Computer Aided Verification, CAV 2012, Proceedings*, pages 378–393, 2012.

[CCM+18] Roberto Cavada, Alessandro Cimatti, Sergio Mover, Mirko Sessa, Giuseppe Cadavero, and Giuseppe Scaglione. Analysis of Relay Interlocking Systems via SMT-based Model Checking of Switched Multi-Domain Kirchhoff Networks. In *2018 Formal Methods in Computer Aided Design, FMCAD 2018*, 2018.

[CGM+98] Alessandro Cimatti, Fausto Giunchiglia, Giorgio Mongardi, Dario Romano, Fernando Torielli, and Paolo Traverso. Formal verification of a railway interlocking system using model checking. *Formal Asp. Comput.*, 10(4):361–380, 1998.

[CGMT15] Alessandro Cimatti, Alberto Griggio, Sergio Mover, and Stefano Tonetta. HyCOMP: An SMT-based model checker for hybrid systems. In *TACAS*, 2015.

[CGMT16] Alessandro Cimatti, Alberto Griggio, Sergio Mover, and Stefano Tonetta. Infinite-state invariant checking with IC3 and

predicate abstraction. *Formal Methods in System Design*, 49(3):190–218, 2016.

[CGSS13] Alessandro Cimatti, Alberto Griggio, Bastiaan Joost Schaafsma, and Roberto Sebastiani. The mathsat5 smt solver. In *TACAS*. Springer Berlin Heidelberg, 2013.

[CMS16] Alessandro Cimatti, Sergio Mover, and Mirko Sessa. From Electrical Switched Networks to Hybrid Automata. In *FM 2016: Formal Methods, Proceedings*, pages 164–181, 2016.

[CMS17] Alessandro Cimatti, Sergio Mover, and Mirko Sessa. SMT-based analysis of switching multi-domain linear Kirchhoff networks. In *2017 Formal Methods in Computer Aided Design, FMCAD 2017*, pages 188–195, 2017.

[CMT11a] Alessandro Cimatti, Sergio Mover, and Stefano Tonetta. Hydi: A language for symbolic hybrid systems with discrete interaction. In *Euromicro SEAA*, pages 275–278, 2011.

[CMT11b] Alessandro Cimatti, Sergio Mover, and Stefano Tonetta. Hydi: A language for symbolic hybrid systems with discrete interaction. In *37th EUROMICRO Conference on Software Engineering and Advanced Applications, SEAA 2011.*, pages 275–278. IEEE Computer Society, 2011.

[CMT14] Alessandro Cimatti, Sergio Mover, and Stefano Tonetta. Quantifier-free encoding of invariants for hybrid systems. *Formal Methods in System Design*, 45(2):165–188, 2014.

[Com76] Comitato Elettrico Italiano. UNIFER-CEI S 461: sigle e segni grafici per gli schemi dei circuiti elettrici degli impianti di seg-

nalamento ferroviario. Standard, Comitato Elettrico Italiano, 1976.

[DDM04] Thao Dang, Alexandre Donzé, and Oded Maler. Verification of analog and mixed-signal circuits using hybrid system techniques. In *FMCAD 2004*, 2004.

[DMBF13] Arash M. Dizqah, Alireza Maheri, Krishna Busawon, and Peter Fritzson. Modeling and simulation of a combined solar and wind energy system using openmodelica. In *5th OpenModelica / 7th MODPROD Annual Workshops*, pages 1–20, 2 2013.

[EOS97] H. Elmqvist, M. Otter, and C. Schlegel. Physical modeling with modelica and dymola and real-time simulation with simulink and real time workshop. In *Matlab User Conference*, 1997.

[Eri04] Lars-Henrik Eriksson. Using formal methods in a retrospective safety case. In *Computer Safety, Reliability, and Security, SAFECOMP 2004, Proceedings*, pages 31–44, 2004.

[FGD+11] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. SpaceEx: Scalable Verification of Hybrid Systems. In *CAV*, pages 379–395, 2011.

[FHM17] Alessandro Fantechi, Anne Elisabeth Haxthausen, and Hugo Daniel Macedo. Compositional verification of interlocking systems for large stations. In *Software Engineering and Formal Methods, SEFM 2017, Proceedings*, pages 236–252, 2017.

[FKRM06] Goran Frehse, Bruce H. Krogh, Rob A. Rutenbar, and Oded Maler. Time domain verification of oscillator circuit properties. *Electr. Notes Theor. Comput. Sci.*, 153(3):9–22, 2006.

[FMQ+15] Nathan Fulton, Stefan Mitsch, Jan-David Quesel, Marcus Völp, and André Platzer. Keymaera X: an axiomatic tactical theorem prover for hybrid systems. In *CADE*, 2015.

[Fri14] Peter Fritzson. *Principles of object-oriented modeling and simulation with Modelica 3.3: a cyber-physical approach*. John Wiley & Sons, 2014.

[GKC13a] Sicun Gao, Soonho Kong, and Edmund M. Clarke. dreal: An SMT solver for nonlinear theories over the reals. In *CADE-24*, 2013.

[GKC13b] Sicun Gao, Soonho Kong, and Edmund M. Clarke. Satisfiability modulo odes. In *Formal Methods in Computer-Aided Design, FMCAD 2013, Portland, OR, USA, October 20-23, 2013*, pages 105–112, 2013.

[GM15] Marco Gario and Andrea Micheli. pySMT: a Solver-Agnostic Library for Fast Prototyping of SMT-Based Algorithms. In *SMT Workshop*, 2015.

[GNO17] GNOME. Dia. `https://gitlab.gnome.org/GNOME/dia`, 2017.

[HCC+00] Vicky Hartonas-Garmhausen, Sérgio Vale Aguiar Campos, Alessandro Cimatti, Edmund M. Clarke, and Fausto Giunchiglia. Verification of a safety-critical railway interlocking system with real-time constraints. *Sci. Comput. Program.*, 36(1):53–64, 2000.

[Hen96] Thomas A. Henzinger. The theory of hybrid automata. In *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, 1996*, 1996.

[HHP17] Linh Vu Hong, Anne Elisabeth Haxthausen, and Jan Peleska. Formal modelling and verification of interlocking systems featuring sequential release. *Sci. Comput. Program.*, 133:91–115, 2017.

[HKB11] Anne Elisabeth Haxthausen, Andreas A. Kjær, and Marie Le Bliguet. Formal development of a tool for automated modelling and verification of relay interlocking systems. In *FM 2011: Formal Methods, Proceedings*, pages 118–132, 2011.

[HØ16] Anne Elisabeth Haxthausen and Peter H. Østergaard. On the use of static checking in the verification of interlocking systems. In *Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications - 7th International Symposium, ISoLA 2016, Proceedings, Part II*, pages 266–278, 2016.

[Jan11] Klaus Janschek. *Mechatronic systems design: methods, models, concepts*. Springer Science & Business Media, 2011.

[KGCC15] Soonho Kong, Sicun Gao, Wei Chen, and Edmund M. Clarke. dReach: δ-reachability analysis for hybrid systems. In *TACAS*, 2015.

[LAH+15] Hyun-Sek Lukas Lee, Matthias Althoff, Stefan Hoelldampf, Markus Olbrich, and Erich Barke. Automated generation of hybrid system models for reachability analysis of nonlinear analog circuits. In *ASP-DAC 2015*, 2015.

[LFW+16] Peter Gorm Larsen, John S. Fitzgerald, Jim Woodcock, Peter Fritzson, Jörg Brauer, Christian Kleijn, Thierry Lecomte, Markus Pfeil, Ole Green, Stylianos Basagiannis, and Andrey Sadovykh. Integrated tool chain for model-based design of cyber-physical systems: The INTO-CPS project. In *2016 2nd International Workshop on Modelling, Analysis, and Control of Complex CPS, CPS Data 2016, Vienna, Austria, April 11, 2016*, pages 1–6, 2016.

[Mat18] The Mathworks. Simscape eletrical. `http://it.mathworks.com/help/physmod/sps/index.html`, 2018. Accessed: 2018-11-26.

[Mea97] A. Massarini and et al. Analysis of networks with ideal switches by state equations. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 1997.

[MF16] Stefano Minopoli and Goran Frehse. SL2SX Translator: From Simulink to SpaceEx Models. In *HSCC 2016*, pages 93–98, 2016.

[MMBC11] Karthik Manamcheri, Sayan Mitra, Stanley Bak, and Marco Caccamo. A step towards verification and synthesis from simulink/stateflow models. In *HSCC 2011*, pages 317–318, 2011.

[Mun97] James R Munkres. *Analysis on manifolds*. Westview Press, 1997.

[Pry01] John D Pryce. A simple structural analysis method for daes. *BIT Numerical Mathematics*, 41(2):364–394, 2001.

[Ria08] Ricardo Riaza. *Differential-algebraic systems analytical aspects and circuit applications*. World Scientific, Singapore, SG, 2008.

[SAE11] SAE International. AIR 6110 - Contiguous Aircraft/System Development Process Example, 2011.

[Ska01] D. L. Skaar. Using the superposition method to formulate the state variable matrix for linear networks. *IEEE Transactions on Education*, 44(4), Nov 2001.

[Tiw12] A. Tiwari. HybridSAL Relational Abstracter. In *CAV*, 2012.

[YONS16] Hiroki Yoshikawa, Takatsugu Oda, Kenichiro Nonaka, and Kazuma Sekiguchi. Modeling and simulation for leg-wheel mobile robots using modelica. In *The First Japanese Modelica Conferences*, number 124, 2016.

[ZSS12] Yan Zhang, Sriram Sankaranarayanan, and Fabio Somenzi. Piecewise linear modeling of nonlinear devices for formal verification of analog circuits. In *FMCAD*, pages 196–203, 2012.

[ZTB06] Mohamed H. Zaki, Sofiène Tahar, and Guy Bois. Formal verification of analog and mixed signal designs: Survey and comparison. In *2006 IEEE North-East Workshop on Circuits and Systems*, pages 281–284, June 2006.

Appendix A

Library of components

We provide several details on the library components that we use in the models presented in this thesis. Table A.1 shows the physical dimension of the effort and flow variables in several physical domains.

Physical domain	**Effort**	**Flow**
Electrical	Potential	Current
Hydraulic	Pressure	Volumetric flow-rate
Gas	Pressure	Mass flow-rate
Mechanical translational	Linear velocity	Force
Mechanical rotational	Rotational velocity	Torque
Thermal	Temperature	Heat flow
Magnetic	Magnetomotive force	Magnetic flux

Table A.1: Dimension of the effort and flow variables in different physical domains.

A.1 Algebraic components

We start from the description of algebraic components that does not contain differential behaviors.

A.1.1 Electrical reference

The electrical *reference* of Figure A.1, also called *ground*, is a one-terminal component with one operating mode. The terminal is an electrical terminal. The component provides a reference potential to a node of the network. This reference is conventionally equal to zero $v_- = 0$. There are no additional constraints on the terminal current that is determined by the mutual-interaction of the component with the network.

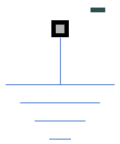


Figure A.1: Electrical reference.

A.1.2 Electrical current source

The electrical *current source* of Figure A.2 is a two-terminal component with one operating mode. The terminals are electrical terminals. The component forces the current i_- on its network branch to be identically equal to a continuous function of time provided by the designer as a parameter of the component. Since the current i_- drives the network with a user-defined time-evolution, we say that i_- is an *input* variable of the network. An additional constraint $i_+ + i_- = 0$ describes the *current conservation law* through the component: the current entering one terminal equals the current exiting the other terminal. No equations about the potential variables are defined because the voltage drop of the component is determined by the mutual-interaction of the component with the network.

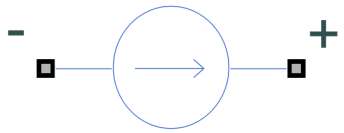


Figure A.2: Electrical current source.

A.1.3 Electrical voltage source

The electrical *voltage source* of Figure A.3 is a two-terminal component with one operating mode. The terminals are electrical terminals. The component forces the voltage drop $v_+ - v_-$ between the terminals to be identically equal to a continuous function of time provided by the designer as a parameter of the component. We call V the voltage drop $v_+ - v_-$ and we say that V is an *input* variable of the network because it drives the network with a user-defined time-evolution. An additional constraint $i_+ + i_- = 0$ describes the *current conservation law* through the component: the current entering one terminal equals the current exiting the other terminal. No equations about the current variables are defined because they are determined by the mutual-interaction of the component with the network.

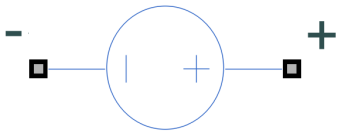


Figure A.3: Electrical voltage source.

A.1.4 Electrical resistor

The electrical *resistor* shown in Figure A.4 is a two-terminal component with one operating mode. The terminals are electrical terminals. The resistor implements the *Ohm's law* $v_+ - v_- = r * i_+$: the voltage drop

$v_+ - v_-$ between the terminals is directly proportional to the current i_+ through the component with a positive proportionality constant r called resistance. The additional constraint $i_+ + i_- = 0$ describes the *current conservation law* through the component. Currents and potentials of the terminal are determined by the mutual-interaction of the component with the network.

Figure A.4: Electrical resistor.

A.1.5 Hydraulic reservoir

Figure A.5 shows the hydraulic *reservoir* component. It is a one-terminal component with one operating mode. The terminal is an hydraulic terminal. The reservoir represents an ideal buffer for the fluid that exceeds the network capacity. The component provides a reference linear velocity to a node of the network. This reference is conventionally equal to zero $p_A = 0$. There are no additional constraints on the terminal flow-rate that is determined by the mutual-interaction of the component with the network.

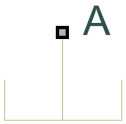


Figure A.5: Hydraulic reservoir.

A.1.6 Hydraulic flow-rate pump

The hydraulic *flow-rate pump* of Figure A.6 is a two-terminal component with one operating mode. The terminals are hydraulic terminals. The

component forces the flow-rate f_T on its network branch to be identically equal to a continuous function of time provided by the designer as a parameter of the component. Since the flow-rate f_T drives the network with a user-defined time-evolution, we say that f_T is an *input* variable of the network. An additional constraint $f_T + f_P = 0$ describes the *flow-rate conservation law* through the component: the flow-rate entering one terminal equals the flow-rate exiting the other terminal. No equations about the pressure variables are defined because the pressure drop of the component is determined by the mutual-interaction of the component with the network.

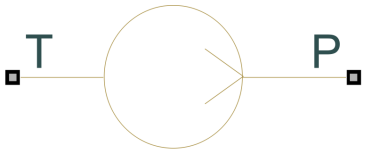


Figure A.6: Hydraulic flow rate pump.

A.1.7 Hydraulic pressure pump

The hydraulic *pressure pump* of Figure A.7 is a two-terminal component with one operating mode. The terminals are hydraulic terminals. The component forces the pressure drop $p_P - p_T$ between the terminals to be identically equal to a continuous function of time provided by the designer as a parameter of the component. We call P the voltage drop $p_P - p_T$ and we say that P is an *input* variable of the network because it drives the network with a user-defined time-evolution. An additional constraint $f_P + f_T = 0$ describes the *flow-rate conservation law* through the component: the flow-rate entering one terminal equals the flow-rate exiting the other terminal. No equations about the flow-rate variables are defined because they are determined by the mutual-interaction of the component with the network.

Figure A.7: Hydraulic pressure pump.

A.1.8 Hydraulic pipeline

The hydraulic *pipeline* shown in Figure A.8 is a two-terminal component with one operating mode. The terminals are hydraulic terminals. The pipeline implements the law $p_A - p_B = r * f_A$ that is similar to the *Ohm's law* of the electrical domain: the pressure drop $p_A - p_B$ between the terminals is directly proportional to the flow-rate f_A through the component with a positive proportionality constant r called resistance. The additional constraint $f_A + f_B = 0$ describes the *flow-rate conservation law* through the component. Flow-rates and pressures of the terminals are determined by the mutual-interaction of the component with the network.

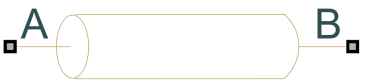


Figure A.8: Hydraulic pipe.

A.1.9 Mechanical reference

The mechanical *reference* of Figure A.9, is a one-terminal component with one operating mode. The terminal is an mechanical-translational terminal. The component provides a reference linear velocity to a node of the network. This reference is conventionally equal to zero $v_A = 0$. There are no additional constraints on the terminal force that is determined by the mutual-interaction of the component with the network.

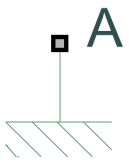


Figure A.9: Mechanical reference.

A.1.10 Mechanical force source

The mechanical *force source* of Figure A.10 is a two-terminal component with one operating mode. The terminals are mechanical translational terminals. The component forces the force f_R on its terminal to be identically equal to a continuous function of time provided by the designer as a parameter of the component. Since the force f_R drives the network with a user-defined time-evolution, we say that f_R is an *input* variable of the network. No equations about the linear velocity variable is defined because it is determined by the mutual-interaction of the component with the network.

Figure A.10: Mechanical force source.

A.2 Algebraic switching components

A.2.1 Electrical two-way switch

The electrical *two-way switch* shown in Figure A.11 is a two-terminal component with two operating modes open and closed. The terminals are electrical terminals. The two-way switch implements the *open-circuit* law $i_- = 0$ in the **open** mode, and the *short-circuit* law $v_+ - v_- = 0$ in the

closed mode. The additional constraint $i_+ + i_- = 0$ describes the *current conservation law* through the component.

Figure A.11: Electrical two-way switch.

A.2.2 Electrical linear diode

The electrical *linear diode* shown in Figure A.12 is a two-terminal component. The terminals are electrical terminals. The diode has two operating modes *direct reverse*. It is in the direct mode when the voltage drop is non-negative (i.e. $v_+ - v_- \geq 0$), otherwise the diode is in the reverse mode. The linear diode implements the *Ohm's law* $v_+ - v_- = r * i_+$ in the **direct** mode, and the *open-circuit* law $i_- = 0$ in the **reverse** mode. The resistance parameter r is almost zero because the component is similar to a short circuit in the direct mode. The additional constraint $i_+ + i_- = 0$ describes the *current conservation law* through the component.

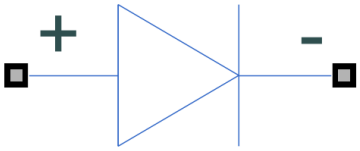


Figure A.12: Electrical linear diode.

A.2.3 Hydraulic two-way valve

The hydraulic *two-way valve* of Figure A.13 is a two-terminal component with two operating mode **open** and **closed**. The terminals are hydraulic terminals. When the valve is open, it connects its terminals allowing the

fluid to move through the component without any pressure loss $p_A - p_B = 0$ (i.e. it is similar to a short-circuit between two terminals in the electrical domain). When the valve is closed, it stops the fluid ($f_A = 0$) regardless of the pressure difference applied by the circuit to the valve (i.e. it is equivalent to an open-circuit in the electrical domain). The additional constraint $f_A + f_B = 0$ describes the *flow-rate conservation law* through the component.

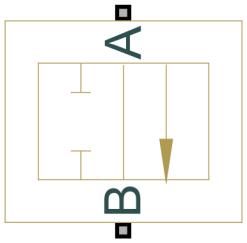


Figure A.13: Hydraulic 2-way valve.

A.2.4 Hydraulic three-way valve

The hydraulic *three-way valve* of Figure A.14 is a three-terminal component with three operating mode **left**, **center**, and **right**. The terminals are hydraulic terminals. In every mode the valve establishes at most one direct connection between two terminals, blocking the liquid flow through the other terminals. When the valve is in the left mode, it connects the A and P terminals with zero pressure drop ($p_A - p_P = 0$), and closes the T terminal ($f_T = 0$). When the valve is in the center mode, closes all the terminals ($f_A = 0 \wedge f_P = 0 \wedge f_T = 0$). When the valve is in the right mode, it connects the A and T terminals with zero pressure drop ($p_A - p_T = 0$), and closes the P terminal ($f_P = 0$). The additional constraint $f_A + f_P + f_T = 0$ describes the *flow-rate conservation law* through the component.

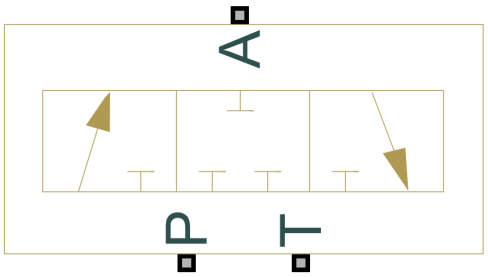


Figure A.14: Hydraulic 3-way valve.

A.2.5 Hydraulic four-way valve

The hydraulic *four-way valve* of Figure A.15 is a four-terminal component with three operating mode **left**, **center**, and **right**. The terminals are hydraulic terminals. In every mode the valve establishes at most two direct connections between pairs of terminals, blocking the liquid flow through the other terminals. When the valve is in the left mode, it connects the A and P terminals with zero pressure drop ($p_A - p_P = 0 \wedge f_A + f_P = 0$), and it connects the B and T terminals with zero pressure drop ($p_B - p_T = 0 \wedge f_B + f_T = 0$). When the valve is in the center mode, it closes all the terminals ($f_A = 0 \wedge f_B = 0 \wedge f_P = 0 \wedge f_T = 0$). When the valve is in the right mode, it connects the A and T terminals with zero pressure drop ($p_A - p_T = 0 \wedge f_A + f_T = 0$), and it connects the B and P terminals with zero pressure drop ($p_B - p_P = 0 \wedge f_B + f_P = 0$). The additional constraint $f_A + f_B + f_P + f_T = 0$ describes the *flow-rate conservation law* through the component.

A.2.6 Hydraulic isolation valve

The hydraulic *isolation valve* shown in Figure A.16 is a two-terminal component. The terminals are hydraulic terminals. The component has two operating modes *direct reverse*. The isolation valve is in the direct mode when the pressure drop is non-negative (i.e. $p_A - p_B \geq 0$), otherwise the

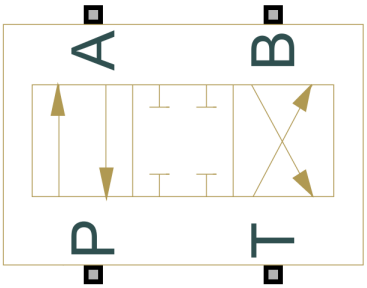


Figure A.15: Hydraulic 4-way valve.

isolation valve is in the reverse mode. The isolation valve implements the pipeline law $p_A - p_B = r * f_A$ in the **direct** mode, and the *open-valve* law $f_A = 0$ in the **reverse** mode. The additional constraint $f_A + f_B = 0$ describes the *flow-rate conservation law* through the component.

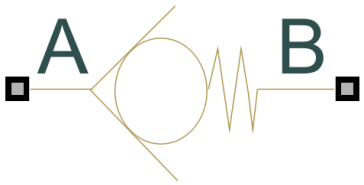


Figure A.16: Hydraulic isolation valve.

A.2.7 Hydraulic fuse

Figure A.17

Figure A.17: Hydraulic fuse.

A.3 Differential components

A.3.1 Electrical capacitor

The electrical *capacitor* of Figure A.18 is a two-terminal component with one operating mode. The terminals are electrical terminals. The capacitor stores energy, or equivalently charge, in the form of electric field. The *charge law* of the capacitor is $c * \frac{dV}{dt} = I$, where V is a shortcut for the voltage drop $v_+ - v_-$ between the terminals, and I is a shortcut for the current i_+ through the positive terminal. This law says that the voltage drop V of the capacitor increases when the current I enters the positive terminal with a positive proportionality constant $1/c$, where the parameter c is called *capacitance*. In order to know the actual voltage drop of the capacitor, we need to integrate its differential law along the time starting from an initial condition. The additional constraint $i_+ + i_- = 0$ describes the *current conservation law* through the component.

Figure A.18: Electrical capacitor.

A.3.2 Electrical inductor

The electrical *inductor* of Figure A.19 is a two-terminal component with one operating mode. The terminals are electrical terminals. The inductor stores energy in the form of magnetic field. The *induction law* of the inductor is $l * \frac{dI}{dt} = V$, where I is a shortcut for the current i_+ through the positive terminal, and V is a shortcut for the voltage drop $v_+ - v_-$ between the terminals. This law says that the current I of the capacitor increases when the voltage drop V with a positive proportionality constant

$1/l$, where the parameter l is called *inductance*. In order to know the actual current of the inductor, we need to integrate its differential law along the time starting from an initial condition. The additional constraint $i_+ + i_- = 0$ describes the *current conservation law* through the component.

Figure A.19: Electrical inductor.

A.3.3 Hydraulic tank

The hydraulic *tank* of Figure A.20 is a one-terminal component with two operating mode **full** and **not-full**. The terminal is an hydraulic terminal. The tank stores energy in the form of stored liquid. When the liquid enters the tank, the internal liquid height and internal energy increases. We need to integrate the differential law of the tank starting from an initial liquid height to compute the current liquid level. When the tank is **full** it cannot accept further incoming liquid from its terminal, otherwise the liquid level varies according to the rate-flow through the terminal.

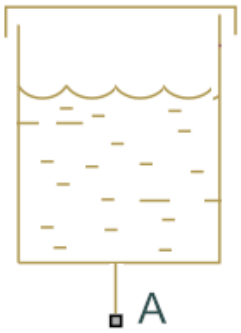


Figure A.20: Hydraulic tank.

A.3.4 Hydraulic accumulator

The hydraulic *accumulator* of Figure A.21 is a one-terminal component with one operating mode. The terminal is an hydraulic terminal. The accumulator stores energy in the form of internal pressure. The internal pressure increases when the liquid enters the accumulator. We need to integrate the differential law of the accumulator along the time, starting from a known initial condition, to compute its current pressure.

Figure A.21: Hydraulic accumulator.

A.3.5 Hydro-mechanical double-acting cylinder

The hydro-mechanical *double-acting cylinder* of Figure A.22 is a three-terminals component with three modes **left**, **center**, and **right**. The terminals A and B are hydraulic terminals, while the terminal R is a mechanical translational terminal. The modes of the component depend on the position of its rod: when the rod is at left/right end-of-stroke the component is in the left/right mode, otherwise it is in the center mode when the rod has an intermediate position. The cylinder works as power transducer from the hydraulic to the mechanical domain. The hydraulic circuit pumps in and out fluid at some pressure from the two hydraulic terminals of the cylinder chambers. When the rod is in an intermediate position of the chambers, the pressures in the chambers create two opposite mechanical forces on the cylinder rod that sum up with the force applied by the external load applied to the mechanical terminal of the cylinder. The net force on the rod determines its direction and the flow rates through the chambers, that in turn determines the linear velocity of the rod. The

mechanical terminal is integral with the rod, so they have the same linear velocity. When the rod reaches its left or right end-of-stroke the hydraulic flow rate through the chambers is blocked (i.e. the cylinder behaves like a closed valve). The position of the rod is a state variable of the system because we need to track it to determine the global state of the circuit. The differential law that governs the position of the rod relates the rod velocity (i.e. the first derivative of the position) and the flow rate of the hydraulic circuit. This flow rate globally depends on the configuration of the system and on the external stimuli that come from the pump flow rate and from the load force (the input variables).

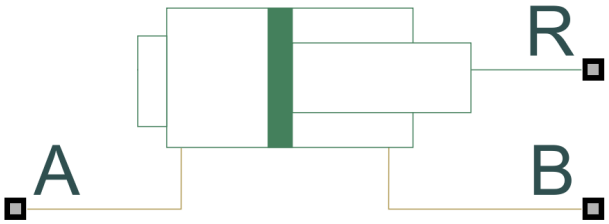


Figure A.22: Double-acting cylinder.

Appendix B

Background on Linear Systems

We represent a system of linear equations using the matrix notation $\vec{A}\vec{x} = \vec{b}$, where $\vec{A} \in \mathbb{R}^{n \times n}$ and $\vec{b} \in \mathbb{R}^n$, and $\vec{x}$ is a n-dimensional vector of real-valued variables. We recall the linear algebra results we will use in the rest of the paper and refer the reader to the book [Axl97] for a detailed treatment.

A system is *homogeneous* if $\vec{b} = \vec{0}$ (i.e., $\vec{A}\vec{x} = 0$). An homogeneous system always admits at least the trivial solution $\bar{\vec{w}} = \vec{0}$.

Given a solvable system $\vec{A}\vec{w} = \vec{b}$, its *general* solution is $\bar{\vec{w}} = \bar{\vec{w}}_p + \bar{\vec{w}}_h$, where $\bar{\vec{w}}_p$ is a *particular* solution of the *inhomogeneous* system $\vec{A}\vec{w} = \vec{b}$ and $\bar{\vec{w}}_h$ is the *homogeneous* solution of the homogeneous system $\vec{A}\vec{w} = \vec{0}$. The existence of the *particular* solution $\bar{\vec{w}}_p$ guarantees the *existence* of at least one solution $\bar{\vec{w}}$.

Definition 11 ((Uniquely)/ solvable linear system)**.** *A linear system $\vec{A}\,\vec{w} = \vec{b}$ is called (uniquely)/ solvable linear system if there exists (a unique)/an assignment $\bar{\vec{w}}$ to the variables $\vec{w}$ that satisfies the system.*

Lemma 3 (Homogeneity)**.** *Let $\vec{A}\,\vec{w} = \vec{b}$ be a linear system. The system $\vec{A}\vec{0} = \vec{b}$ is solvable for any matrix $\vec{A}$ iff $\vec{b} = \vec{0}$ is the zero vector.*

Consequently, the homogeneous system $\vec{A}\,\vec{w} = \vec{0}$ always admits at least the zero solution $\bar{\vec{w}} = \vec{0}$.

Theorem 3 (General solution)**.** *Given a solvable linear system* $\vec{A}\vec{w} = \vec{b}$*, its* general *solution is* $\bar{\vec{w}} = \bar{\vec{w}}_p + \bar{\vec{w}}_h$*, where* $\bar{\vec{w}}_p$ *is any* particular *solution of the inhomogeneous system* $\vec{A}\vec{w} = \vec{b}$ *and* $\bar{\vec{w}}_h$ *is a* homogeneous *solution of the homogeneous system* $\vec{A}\vec{w} = \vec{0}$*.*

The existence of the *particular* solution $\bar{\vec{w}}_p$ guarantees the *existence* of at least one solution $\bar{\vec{w}}$. The *uniqueness* of such $\bar{\vec{w}}$ depends on the properties of the *homogeneous* solution $\bar{\vec{w}}_h$. $\bar{\vec{w}}_h$ belongs to a family of (potentially infinite) solutions. For instance, when such family reduces to the singleton $\{\vec{0}\}$, the solution $\bar{\vec{w}}$ of $\vec{A}\vec{w} = \vec{b}$ is guaranteed to be unique, that is for every component of the vector $\bar{\vec{w}}$ exists a unique assignment that satisfies $\vec{A}\vec{w} = \vec{b}$.

Corollary 1 (Component uniqueness)**.** *Given a solvable linear system* $\vec{A}\vec{w} = \vec{b}$*, its* general *solution* $\bar{\vec{w}}$ *admits a unique assignment to the* j*-th component* w_j *iff there exists no a homogeneous solution* $\bar{\vec{w}}_h$ *with* j*-th component different from zero.*

Given a linear system, we are interested in studying: i) the existence of a *particular* solution to guarantee the existence of a *general* solution, and ii) the shape of the family of *homogeneous* solutions, in order to guarantee the uniqueness of a specific component of the *general* solution.

Lemma 4 (Linearity)**.** *Let* $\vec{A}\,\vec{w} = \vec{b}_1$*, ...,* $\vec{A}\,\vec{w} = \vec{b}_n$ *be* n *distinct linear systems and* $z_1, ..., z_n \in \mathbb{R}$ n *real variables. The systems* $\vec{A}\,\vec{w} = \vec{b}_1$*, ...,* $\vec{A}\,\vec{w} = \vec{b}_n$ *are all solvable iff the system* $\vec{A}\,\vec{w} = \vec{b}_1\,z_1 + ... + \vec{b}_n\,z_n$ *is solvable for all values of the variables* z_1*, ...,* z_n*.* □

Lemma 4 (Linearity)**.** *Let* $\vec{A}\vec{w} = \vec{b}_1$*, ...,* $\vec{A}\vec{w} = \vec{b}_n$ *be* n *linear systems and* $z_1, ..., z_n \in \mathbb{R}$ n *real variables. The systems* $\vec{A}\vec{w} = \vec{b}_1$*, ...,* $\vec{A}\vec{w} = \vec{b}_n$ *are solvable iff the system* $\vec{A}\vec{w} = \vec{b}_1\,z_1 + ... + \vec{b}_n\,z_n$ *is solvable for every assignment to the variables* z_1*, ...,* z_n*.*

If we call $\bar{\vec{w}}_{p_1}, ..., \bar{\vec{w}}_{p_n}$ the particular solutions of the systems $\vec{A}\vec{w} = \vec{b}_1$, ..., $\vec{A}\vec{w} = \vec{b}_n$, respectively, then the particular solution $\bar{\vec{w}}_p$ of the system $\vec{A}\vec{w} = \vec{b}_1\, z_1 + ... + \vec{b}_n\, z_n$ is the linear combination $\bar{\vec{w}}_p = \bar{\vec{w}}_{p_1}\, z_1 + ... + \bar{\vec{w}}_{p_n}\, z_n$ of the $\bar{\vec{w}}_{p_i}$ weighted by the coefficients z_i.
Known the coefficient vectors $\vec{b}_i$, we can see the general solution $\bar{\vec{w}}$ as described by (at least) a linear function $f(z_1, ..., z_n)$ of the variables z_i. Again, the uniqueness of such linear function depends on the shape of the family of homogeneous solutions of the system $\vec{A}\vec{w} = \vec{b}_1\, z_1 + ... + \vec{b}_n\, z_n$.

Appendix C

Proofs

Theorem 1 (Implicit Function Theorem)**.** *Let m, n, l be positive integers. Let $F : \mathbb{R}^{m+n} \to \mathbb{R}^l$ be a homogeneous implicit linear function $F(\vec{w}, \vec{z}) := \vec{A}\vec{w} + \vec{B}\vec{z} = \vec{0}$, where $\vec{w} \in \mathbb{R}^{m\times 1}$, $\vec{z} \in \mathbb{R}^{n\times 1}$, $\vec{A} \in \mathbb{R}^{l\times m}$, and $\vec{B} \in \mathbb{R}^{l\times n}$. Let $\vec{b}_i$ be the i-th column vector of the matrix $\vec{B}$, where $i \in \{1, ..., n\}$. Let w_j be the j-th variable of $\vec{w}$, where $j \in \{1, ..., m\}$. The following two conditions hold:*

1. *consistency condition: for all $0 \leq i \leq n$, the linear system $\vec{A}\vec{w} = -\vec{b}_i$ is solvable, and*

2. *determinism condition: the linear system $\vec{A}\vec{w} = \vec{0}$ does not admit any homogeneous solution $\bar{\vec{w}}_h$ such that its j-th component w_j is different from zero*

iff there exists a unique linear function $f_j : \mathbb{R}^n \to \mathbb{R}^1$ such that $w_j = f_j(\vec{z})$ and $F(w_1, ..., f_j(\vec{z}), ..., w_m, \vec{z}) = \vec{0}$.

Proof. ($\Rightarrow$) We can interpret every system $\vec{A}\vec{w}_i = \vec{b}_i$ as the system $\vec{A}\vec{w} = -\vec{B}\vec{z}$ whose rhs $-\vec{B}\vec{z}$ is obtained as the product $-\vec{b}_i\, z_i$ of the i-th column $\vec{b}_i$ of the matrix $\vec{B}$ and the coefficient $z_i = 1$, namely $\vec{A}\vec{w}_i = -\vec{b}_i 1$. According to the Lemma 4, the first condition guarantees that at least one *particular* solution $\bar{\vec{w}}_p = f(z_1, ..., z_n)$ exists for the system $\vec{A}\vec{w} = -\vec{B}\vec{z}$, for every

assignment to the independent variables $\vec{z}$.
The second condition states that the family of *homogeneous* solutions does not contain any vector whose j-th component w_j is different from zero. Thus, according to the Corollary 1, for every assignment to the variables $\vec{z}$, the solvable system $\vec{A}\vec{w} = -\vec{B}\vec{z}$ admits a general solution $\bar{\vec{w}}$ whose j-th component w_j is uniquely assigned.

($\Leftarrow$) Let us assume that a unique linear function $f_j : \mathbb{R}^n \to \mathbb{R}^1$ exists and that $F(w_1, ..., f_j(\vec{z}), ..., w_m, \vec{z}) = 0$ holds for every assignment to the variables $\vec{z}$.

For any of the n assignments $\vec{z} = (0, 0, ..., 1, ..., 0, 0)$ to the independent variables $\vec{z}$, the implicit function $F(\vec{w}, \vec{z})$ reduces to $\vec{A}\vec{w}_i = -\vec{b}_i 1$. For the assumption that $F(\vec{w}, \vec{z})$ is solvable for every assignment to the independent variables $\vec{z}$, also any of the n $\vec{A}\vec{w}_i = -\vec{b}_i$ is solvable.

For the assignment $\vec{z} = (0, 0, ..., 0, 0)$, the implicit function $F(\vec{w}, \vec{z})$ reduces to $\vec{A}\vec{w} = \vec{0}$, and, according to the Lemma 3 it is solvable for $\bar{\vec{w}} = \vec{0}$, thus for the existence of f_j, $w_j = f_j(\vec{0}) = 0$. For the uniqueness of f_j, no other solutions of $\bar{\vec{w}} = \vec{0}$ exist such that their j-th component w_j takes values different from 0.

□

Lemma 1. *μ is a satisfying model of ψ_{DAE} iff $\mu_{|R}$ is a solution of $\text{DAE}(\mu_{|B})$*

Proof. ($\Rightarrow$) If $\mu \models \psi_{\text{DAE}}$, then $\mu \models Flow_i(\mu_{|B_i})$ for all $c_1 \in \mathcal{C}$, $\mu \models k$ for all $k \in K$. Hence, $\text{DAE}(\mu_{|B})$.

($\Leftarrow$) Let μ' be a solution (defined over the variables R) to $\text{DAE}(m)$, for a $m \in 2^B$. Then, it is easy to see that $\mu'' = \mu' \cup m$ is a model for ψ_{DAE} (where $\mu' \cup m$ denotes the union of the two models).

□

Lemma 5. *Given a network $\mathcal{N}$ and a mode $m \in 2^B$, m belongs to the set represented by the formula $\psi_{con}(B)$ (i.e. $m \models \psi_{con}(B)$) iff the DAE*

DAE(m) *satisfies the condition (1) of Theorem 1.*

Proof. ($\Rightarrow$) Assume that $m \models \psi_{con}(B)$. This means that m satisfies all the quantified formulas $\exists \dot{X}, R.\left(\psi_{\text{DAE}}\left[\delta_{z_i}^{U \cdot X}/U \cdot X\right]\right)$ that can be rewritten as the formulas $\exists \vec{w}, \vec{z}.\left(\text{DAE}(m)\left[\delta_{z_i}^{\vec{z}}/\vec{z}\right]\right)$ that are equivalent to the formulas $\exists \vec{w}, \vec{z}.\left(\vec{A}\vec{w} = -\vec{b_i}\right)$, where $\vec{b_i}$ is the i-th column of the concatenated coefficient matrix $\vec{B}$. This means that the DAE DAE(m) satisfies the condition (1) of the Theorem 1.

($\Leftarrow$) Assume that the DAE DAE(m) satisfies the condition (1) of the Theorem 1. This means that, for the mode m, every system $\vec{A}\vec{w} = -\vec{b_i}$ is solvable, consequently every quantified formula $\exists \vec{w}, \vec{z}.\left(\vec{A}\vec{w} = -\vec{b_i}\right)$ is satisfied from the mode m. Thus, $m \models \psi_{con}(B)$.

□

Lemma 6. *Given a network $\mathcal{N}$ and a mode $m \in 2^B$, m belongs to the set represented by the formula $\psi_{det}(B)$ (i.e. $m \models \psi_{det}(B)$) iff the DAE* DAE(m) *satisfies the condition (2) of the Theorem 1 for all the variables in $\dot{X}$.*

Proof. ($\Rightarrow$) Assume that $m \models \psi_{det}(B)$. This means that m satisfies the quantified formula $\neg\exists \dot{X}, R.\left(\psi_{\text{DAE}}\left[\vec{0}/U\right]\left[\vec{0}/X\right] \wedge \left(\dot{X} \neq \vec{0}\right)\right)$, and equivalently that m does not satisfy the quantified formula $\exists \dot{X}, R.\left(\psi_{\text{DAE}}\left[\vec{0}/U\right]\left[\vec{0}/X\right] \wedge \left(\dot{X} \neq \vec{0}\right)\right)$ that can be rewritten as the formula $\exists \vec{w}, \vec{z}.(\vec{A}\vec{w} = \vec{0} \wedge \bigvee_{l \in \dot{X}}(l \neq 0))$. Since the existential quantification distributes over the logical disjunction, we can rewrite the formula as $\bigvee_{l \in \dot{X}} \exists \vec{w}, \vec{z}.(\vec{A}\vec{w} = \vec{0} \wedge (l \neq 0))$. For the assumption, the mode m does not satisfy any disjunct $\exists \vec{w}, \vec{z}.(\vec{A}\vec{w} = \vec{0} \wedge (l \neq 0))$ for all $l \in \dot{X}$. Thus, the homogeneous system $\vec{A}\vec{w} = \vec{0}$, that always admits the trivial solution, does not admit any solution whose component l is different from zero. Consequently, the DAE DAE(m) satisfies the condition (2) of the Theorem 1 for all the variable in $\dot{X}$.

($\Leftarrow$) Assume that the DAE $\textsc{Dae}(m)$ satisfies the condition (2) of the Theorem 1 for all the variable $l \in \dot{X}$. This means that the homogeneous system $\vec{A}\vec{w} = \vec{0}$ does not admit any a solution whose component $l \neq 0$. Consequently, all the disjunct $\exists \vec{w}, \vec{z}.(\vec{A}\vec{w} = \vec{0} \wedge (l \neq 0))$ are not satisfied by the mode m, thus $m \models \psi_{det}\,(B)$.

□

Lemma 2. *A network $\mathcal{N}$ is consistent iff for all $m \in 2^B$, $m \models \psi_{con}(B)$, and is deterministic iff for all the modes $m \in 2^B$, $m \models \psi_{det}(B)$*

Proof. The proof trivially follows from the application of Lemmas 5 and 6 to all the network modes $m \in 2^B$.

□

Lemma 7. *Let $\mathcal{N}$ be a valid network, $\dot{x}$ a variable in $\dot{X}$, m a mode in 2^B, D the coefficients returned by* $\textsc{GetCoeff}\,(\psi_{\text{DAE}}, X, U, \dot{x}, m)$*, and $f_{\dot{x}} : \mathbb{R}^n \to \mathbb{R}^1$ the unique linear function that reformulates $\dot{x}$ from* $\textsc{Dae}(m)$ *(according to Lemma 4), where $n = |\vec{z}|$.*

The equation $\dot{x} = D\vec{z}$ has the same set of solutions as $\dot{x} = f_{\dot{x}}(\vec{z})$, where $\vec{z} := (U \cdot X)$.

Proof. $\textsc{GetCoeff}$ asserts the formula $\psi_{\text{DAE}} \wedge m$ that, by Lemma 1, encodes $\textsc{Dae}(\mu_{|B})$.

In each iteration of the loop, $\textsc{GetCoeff}$ computes the i-th value of the coefficients vector D. This amounts to find a solution for the system $\vec{A}\vec{w} = -\vec{b_i}$ (encoded at Line 5 of the algorithm by asserting $z_i = 1 \wedge \bigwedge_{l \in (U \cup X) \setminus \{z_i\}} l = 0$). The coefficient $\bar{\vec{w}}^{\dot{x}}_{p_i}$ of the function $f_{\dot{x}}(\vec{z}) := \bar{\vec{w}}^{\dot{x}}_{p_1}\, z_1 + \ldots + \bar{\vec{w}}^{\dot{x}}_{p_n}\, z_n$ (as described in Lemma 4) is the value assigned to $\dot{x}$ in the solution (Line 7).

Hence, at the end of $\textsc{GetCoeff}$, D is the vector of coefficients of the linear function $f_{\dot{x}}(\vec{z})$.

□

Lemma 8. *Let $\mathcal{N}$ be a valid network, $\dot{x}$ a variable in $\dot{X}$, β the set of modes returned by* GetEqMod $(\psi_{\text{DAE}}, X, U, \dot{x}, D)$.

For all $m \in 2^B$, $m \in \beta$ iff the equation $\dot{x} = D\vec{z}$ has the same set of solutions as $\dot{x} = f_{\dot{x}}(\vec{z})$, where $f_{\dot{x}}$ is the unique linear function that reformulates $\dot{x}$ in DAE(m), *and $\vec{z} := (U \cdot X)$.*

Proof. The lemma can be proved by noticing that GetEqMod encodes in a formula the set of all the possible modes that assign the coefficients D to $\dot{x}$. These are also the same coefficients of $f_{\dot{x}}(\vec{z})$, by applying the reasoning done in the proof of Lemma 7.

□

Theorem 2 (Correctness of the reformulation)**.** *Given a valid network $\mathcal{N}$, the hybrid automaton $H^r_{\mathcal{N}}$ is equivalent to the hybrid automaton $H_{\mathcal{N}}$ that defines the network semantics.*

Proof. ($\Rightarrow$) $H^r_{\mathcal{N}}$ and $H_{\mathcal{N}}$ have the same state variables, initial states and transition relation [1]. If $\pi = s_0; s_1; \ldots; s_n$ is a path of $H^r_{\mathcal{N}}$, then π is a path of $H_{\mathcal{N}}$. We prove that π is a path of $H_{\mathcal{N}}$ by induction. Clearly, $s_o \models Init$.
By hypothesis, $s_0; s_1; \ldots; s_{i-1}$, $i \leq k$ is a path of $H_{\mathcal{N}}$.
We prove that $s_{i-1} \xrightarrow{\delta_i} s_i$ is a transition of $H_{\mathcal{N}}$.
If the transition is discrete, $\delta_i = d$, then $\langle s_{i-1}, \delta_i, s_i \rangle \models Trans$.
If the transition is continuous, $\delta_i > 0 \in \mathbb{R}$, then there exists a continuous differentiable function $f : [0, \delta_i] \to \mathbb{R}^{|R^r|}$ such that:

1. $f(0) = s_{i-1|R^r}$ and $f(\delta_i) = s_{i|R^r}$,
2. $s_{i-1} \models Invar^r$ and $s_i \models Invar^r$,
3. $\forall \epsilon \in [0, \delta_i]$, $\langle s_{i-1|B^r}, f(\epsilon), \dot{f}(\epsilon) \rangle \models Flow^r$
4. $\forall \epsilon \in [0, \delta_i]$, $\langle s_{i-1|B^r}, f(\epsilon) \rangle \models Invar^r$.

[1] In practice, when constructing $H^r_{\mathcal{N}}$ we do not explicitly enumerate all the modes $m \in 2^B$. The equivalence between $Trans$ and $Trans^r$ is straightforward.

The function $f : [0, \delta_i] \to \mathbb{R}^{|R^r|}$ is also such that:

1. $s_{i-1} \models Invar$ and $s_i \models Invar$.

   We know that $s_{i-1} \models \psi_Y$. We know that $s_{i-1|\dot{R}^r \cup R^r}$ is a solution for $\psi_{Y,\dot{x}}(s_{i-1|B^r})$ (i.e. every time we have a model in $Invar^r$ we also have a model in $Invar$).

   The same reasoning applies for s_i.

2. $\forall \epsilon \in [0, \delta_i]$, $\langle s_{i-1B^r}, f(\epsilon), \dot{f}(\epsilon) \rangle \models Flow^r$

   By Lemma 7 and Lemma 8, we know that the reformulation $Ref_{\dot{x}}(s_{i-1|B^r})$ is equivalent to $\dot{x} = f_{\dot{x}}(\vec{z})$, for every $\dot{x} \in \dot{X}$.

   Hence, f is a solution to the system of differential equations formed by all the equations $\dot{x} = f_{\dot{x}}(\vec{z})$, for all $\dot{x} \in \dot{X}$, and hence is a solution for $Flow$

3. $\forall \epsilon \in [0, \delta_i], \langle s_{i-1|B}, f(\epsilon) \rangle \models Invar^r$.

   This can be proved again by observing that f is a solution for $Flow$, that the invariants hold for all the possible ϵ in $Invar^r$ and every time we have a model in $Invar^r$ we have also a model in $Invar$.

Then, $s_{i-1} \xrightarrow{\delta_i} s_i$ is a continuous transition in $H_{\mathcal{N}}$.
By induction π is a path of $H_{\mathcal{N}}$.
($\Leftarrow$) If $\pi = s_0; s_1; \ldots; s_n$ is a path of $H_{\mathcal{N}}$, then π is a path of $H^r_{\mathcal{N}}$.
This direction can be proved similarly to ($\Rightarrow$).

□

Acknowledgments

I would like to thank:

- my advisor, Alessandro Cimatti, and my co-advisor, Sergio Mover, for their guidance and constant help. I am grateful for the beautiful way we have done together and for everything they have taught me in the last years;
- the members of the Doctorate Committee, for the time they spent reading my thesis;
- the ICT Secretariat, in particular Francesca and Andrea, for their support;
- my colleagues at University of Trento and Fondazione Bruno Kessler, for their feedback and support;
- my friends in Trento, back home, and all around the globe, for their support and for all the fun times that colored my PhD adventure; and
- my family.