# From Electrical Switched Networks to Hybrid Automata (Extended Version)

Alessandro Cimatti[1], Sergio Mover[2], and Mirko Sessa[1,3]

[1] Fondazione Bruno Kessler, Trento, IT, {cimatti, sessa}@fbk.eu
[2] University of Colorado at Boulder, sergio.mover@colorado.edu
[3] University of Trento, Trento, IT

**Abstract.** In this paper, we propose a novel symbolic approach to automatically synthesize a Hybrid Automaton (HA) from a switched electrical network. The input network consists of a set of physical components interconnected according to some reconfigurable network topology. The underlying model defines a local dynamics for each component in terms of a Differential-Algebraic Equation (DAE), and a set of network topologies by means of discrete switches. Each switch configuration induces a different topology, where the behavior of the system is a Hybrid Differential-Algebraic Equations.
Two relevant problems for these networks are validation and reformulation. The first consists of determining if the network admits an Ordinary Differential Equations (ODE) that describes its dynamics; the second consists of obtaining such ODE from the initial DAE. This step is a key enabler to use existing formal verification tools that can cope with ODEs but not with DAEs.
Since the number of network topologies is exponential in the number of switches, first, we propose a technique based on Satisfiability Modulo Theories (SMT) that can solve the validation problem symbolically, avoiding the explicit enumeration of the topologies. Then, we show an SMT-based algorithm that reformulates the network into a symbolic HA. The algorithm avoids to explicitly enumerate the topologies clustering them by equivalent continuous dynamics.
We implemented the approach with several optimizations and we compared it with the explicit enumeration of configurations. The results demonstrate the scalability of our technique.

## 1 Introduction

Many practical systems feature emerging behaviors from the complex interactions of physical components, that are interconnected according to some reconfigurable topology. Typical examples include hydraulics [25] and electrical power supply networks [22]. The components interact by exchanging energy along the network branches, in a bidirectional fashion, that results in a relational, global model, that depends on the specific system configuration (or mode).

Kirchhoff Networks [13] are a well-known and powerful framework for component-based physical modeling, that allows to cover power-conserving network rules. Dedicated analysis methods, devised for "single-mode" networks, support the validation of some basic sanity properties (e.g. absence of VC-loops, or IL cutsets [23]). Following

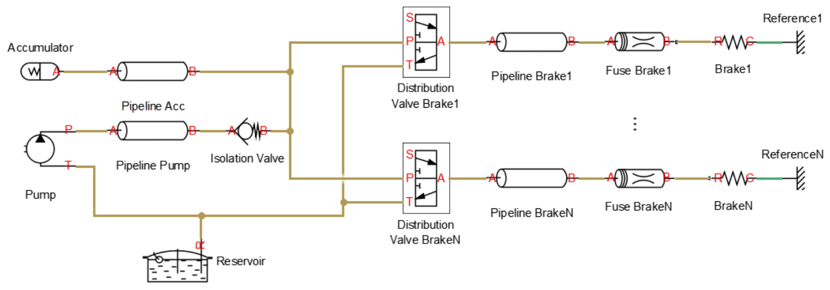


**Fig. 1.** Wheel Braking Systems (WBS) with N braking lines.

the Electronic-Hydraulic analogy [2], it is also possible to analyze interesting classes of hydraulic circuits, such as the WBS in Figure 1.

Unfortunately, dynamic reconfiguration yields networks that are associated with a potentially exponential number of modes. Consider, for example, the simple electrical circuit in Figure 2: depending on the status of the switches and fuses, sixteen configurations are possible, each of which is associated with a suitable set of differential equations. Similar considerations apply, on a larger scale, to the hydraulic circuit from [25] in Figure 1. In this paper, we tackle two key problems. The *network validation* problem consists of showing that the dynamics of the network can be expressible in form of an ODE (in order for it to be amenable to formal verification), and that all the output variables (i.e. variables that should be functionally represented by the state of the network) can be uniquely determined. The *network reformulation* problem consists of converting the network into an equivalent hybrid automaton, in order to enable functional verification of the network. The challenge lies in the fact that, for each discrete configuration, the dynamics of the network is defined by a Differential-Algebraic Equation (DAE). However, the available tools expect an Ordinary Differential Equation (ODE). Although solving the problem for a fixed configuration is rather simple, given a configuration of the switches, the number of configurations is exponential in the number of switches. Thus, an enumerative approach is hardly feasible: one would have to analyze all the possible modes, to rule out the ones that are deemed unfeasible and build a suitable equational model for each of the remaining modes. In practice, such an approach is not feasible, for two main reasons. First, a manual approach is an extremely tedious and error prone task. Second, an enumerative approach may results in an enormous model, that is hardly manageable for verification tools. Such an enumerative approach is in fact applied in [21], to the modeling of a few two-switches circuit.

In this work, we discuss how to automatically reformulate a Linear Electrical Kirchhoff Network into the corresponding Hybrid Automaton with ODE continuous dynamics. We propose a symbolic approach to the network validation problem, and a symbolic algorithm for the reformulation problem. The idea is to aggregate the discrete modes that share the same dynamics, with different variants in the computation. This results in a more efficient reformulation, and a more compact Hybrid Automaton.

The approach is experimentally evaluated on several electrical and hydraulic benchmarks, where we carry out the validation, and compare (the variants of) the proposed symbolic reformulation approach with its enumerative counterpart. The results demonstrate a much greater scalability of the symbolic approach. We also discuss the application of the proposed approach to some benchmarks in the literature [15, 21].

The paper is organized as follows: In Section 2 we describe some background. In Section 3 we formally define the problem at hand. In Section 4 we describe the validation routine, and in Section 5 we describe the reformulation methods. In Section 6 we describe the related work, and in Section 7 we experimentally evaluate the proposed approach. In Section 8 we draw some conclusions and promising research directions. In Appendix we work out a motivating example, report the proofs of the theorems, and present some additional experimental results.

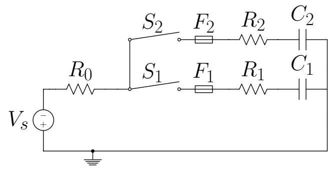


**Fig. 2.** Switched RC Network.

## 2 Background

**Notation.** We use the standard notions of theory, satisfiability, validity, and logical consequence. We restrict to formulas interpreted with the Theory of Linear Real Arithmetic (LRA) [4].

Given a formula in first-order logic $\psi$ and a set of variables $X$, we write $\psi(X)$ to denote that $X$ is the set of free variables in $\psi$. We write $\varphi \models_{\mathcal{T}} \psi$ to denote that the formula $\psi$ is a logical consequence of $\varphi$ in the theory $\mathcal{T}$; when clear from context, we omit $\mathcal{T}$ and simply write $\varphi \models \psi$. An assignment $\mu$ for a set of variables $X$ is the set $\{x \mapsto c \mid x \in X$ and $c$ is a constant$\}$, $\mu_{|X}$ is the projection of all the assignments in $\mu$ only to variables contained in $X$, and $\mu(x)$ is the value assigned to $x$ in $\mu$. We denote with $|X|$ the cardinality of the set $X$. Given a set of real variables $X$, we will use the the notation $\boldsymbol{X}$ to refer to the vector that contains all the variables in $X$ ordered in a lexicographic order. If $X$ is a set of variables, then $X'$ and $\dot{X}$ are the sets obtained by replacing each element $x$ with its primed and dotted version respectively.

**Hybrid Automata.** *Hybrid automata* (HA) [12] represent a system with continuous and discrete dynamics. We use a symbolic representation of hybrid automata, where the discrete locations and transitions are represented symbolically [8].

A *Hybrid automaton* is a tuple $H = \langle D, R, Init, Invar, Trans, Flow\rangle$ where 1) $D$ is the set of discrete variables; 2) $R$ is the set of continuous variables; 3) $Init(D, R)$ represents the set of initial states; 4) $Invar(D, R)$ represents the set of invariant states; 5) $Trans(D, R, D', R')$ represents the set of discrete transitions; 6) $Flow(D, \dot{R}, R)$ represents the flow condition. We assume that all the formulas $Init$, $Invar$, $Trans$ and $Flow$ are quantifier-free and linear. We assume $Invar$ to be of the form $\psi(D) \rightarrow \bigwedge_{p\in P} p(R)$, where $p \in P$ is a predicate, to ensure the convexity of the invariants. We assume $Flow$ to be of the form $\psi(D) \rightarrow \bigwedge_{p\in P} p(R, \dot{R})$, where $p \in P$ is an equality.

In the above definition, $Flow$ may either define a system of Differential-Algebraic Equations (DAEs) or Ordinary Differential Equations (ODEs). We say that the automa-

| Component | Constitutive relation | $X_i$ | $U_i$ | $Y_i$ | Constants |
|---|---|---|---|---|---|
| Voltage source | $v_i = V_s$ | $\emptyset$ | $\{v_i\}$ | $\{i_i, v_i^+, v_i^-\}$ | $V_s$ |
| Current source | $i_i = I_s$ | $\emptyset$ | $\{i_i\}$ | $\{v_i, v_i^+, v_i^-\}$ | $I_s$ |
| Resistor | $v_i = R\, i_i$ | $\emptyset$ | $\emptyset$ | $\{i_i, v_i, v_i^+, v_i^-\}$ | $R$ |
| Capacitor | $i_i = C\, \dot{v}_i$ | $\{v_i\}$ | $\emptyset$ | $\{i_i, v_i^+, v_i^-\}$ | $C$ |
| Inductor | $v_i = L\, \dot{i}_i$ | $\{i_i\}$ | $\emptyset$ | $\{v_i, v_i^+, v_i^-\}$ | $L$ |
| Ground | $v_i^+ = 0$ | $\emptyset$ | $\emptyset$ | $\{i_i, v_i^+\}$ | |

**Table 1.** Constitutive relations and variables of continuous components.

ton has an *ODE dynamics* if, for each assignment $\mu$ to $D$, the conjunct of $\psi(D) \rightarrow \bigwedge_{p \in P} p(R, \dot{R})$ that holds for $\mu$ is a system of ODEs. Otherwise, the automaton has a *DAE dynamics*.

A *state* of a hybrid automaton $H$ is an assignment $s$ to the variables $D \cup R$. Informally, a *run* of the automaton is a sequence of states such that the first state is in the initial states, every state belongs to the invariant states, and each pair of consecutive states either satisfies a discrete transition or follows the solution of the differential equations described in the flow condition. The semantics of the HA is provided in terms of the runs that it accepts.

**Electrical Networks.** An (non-switched) electrical network is formed by the connection of a set of (continuous) components. Without loss of generality, we consider only components with two terminals. A *continuous component* $e_i$ defines two quantities, the voltage (difference of potential) across the two terminals of the component, and the current that flows through the component. We denote with $i_i(t)$ and $v_i(t)$ the current and the voltage of $e_i$. $v_i(t)$ and $i_i(t)$ change continuously in time according to a *constitutive relation*, $\psi_i$, a Differential-Algebraic Equation among $v_i(t)$, $i_i(t)$, and their derivatives $\dot{v}_i, \dot{i}_i$. Furthermore, for each component $e_i$ we consider the variables $v_i^-$ and $v_i^+$, that represent the value of the potential at the two terminals of $e_i$. These variables are connected to $v_i$ through the *voltage equation* $v_i = v_i^+ - v_i^-$.

We denote with $V_i$ the set of variables of $e_i$. We further partition $V_i$ into the following subsets: 1) $X_i := \{x \mid \dot{x}\ \textit{appears in}\ \psi_i\}$ is the set of *state* variables (their derivatives appear in the constitutive relation of $e_i$); 2) $U_i$ is the set of *input* variables, which depends on the component type; 3) $Y_i$ is the set of *output* variables, which depends on the component type. The sets $X_i$, $U_i$, $Y_i$ are disjoint, and $V_i = X_i \cup U_i \cup Y_i$. $\dot{X}_i$ is the set of first derivatives of $X_i$. In Table 1 we report the constitutive relations for the electrical components considered in this paper and their sets of variables. Furthermore, we say that a component is *active* if it has at least a state or input variable, and *passive* otherwise.

The connection of components terminals is represented by a directed graph: an oriented edge represents a component and a node represents the connection of components terminals. As usual, the orientation of the edges is chosen arbitrarily assuming a reference direction for the current.

**Definition 1 (Electrical network).** *An* electrical network *[5, 24] is a directed graph $G = \langle N, E, \eta \rangle$, where 1) $N$ is a set of nodes; 2) $E$ is a set of components; 3) $\eta : E \mapsto N \times N$ defines the directed edges between nodes.*

Let $E_n^{in} = \{e \mid e \in E \text{ and } (n_1, n) = \eta(e)\}$ and $E_n^{out} = \{e \mid e \in E \text{ and } (n, n_1) = \eta(e)\}$ be the sets of incoming and outgoing edges of the node $n$. Additionally, let $P_n = \bigcup_{e_i \in E_n^{in}} v_i^+ \cup \bigcup_{e_i \in E_n^{out}} v_i^-$ denote the set of all the potentials of the components incident on the node $n$, considering also the direction sign imposed by the edge orientation. The connection of the components is described by the *Kirchhoff Current Law* ($KCL$) and the *Kirchhoff Voltage Law* ($KVL$):

$$KCL_G := \bigcup_{n \in N} \left( \sum_{c_i \in E_n^{in}} i_i - \sum_{c_i \in E_n^{out}} i_i = 0 \right) \quad KVL_G := \bigcup_{n \in N} \bigcup_{p_1, p_2 \in P_n} (p_1 = p_2)$$

**Definition 2 (Differential-Algebraic Equation of a network).** *Given a network $G = \langle N, E, \eta \rangle$, its associated DAE, called $DAE_G$, is defined by the set of constitutive equations $\{\psi_i \mid e_i \in E\}$, the set of voltage equations $v_i = v_i^+ - v_i^-$, and the sets of algebraic equations $KCL_G$ and $KVL_G$.*

While there exist several equivalent DAE systems to represent an electrical network, we basically use the one obtained applying the Node Tableau Analysis (NTA) [24].

We extend the notation used to specify the component's variables and their partitions to a network $G$. Hence, we have the sets $V_G := \bigcup_{e_i \in E} V_i$, $X_G := \bigcup_{e_i \in E} X_i$, $U_G := \bigcup_{e_i \in E} U_i$, $Y_G := \bigcup_{e_i \in E} Y_i$. A *state* of the network is given by an assignment $\mu$ to all the variables $V_G$. A state $\mu$ is a *consistent initial value* for $DAE_G$ if $DAE_G$ has a solution for $\mu$ (i.e. if replacing all the variables with the assigned constants in $\mu$ the resulting system of algebraic equations has a solution). A variable $y \in Y_G$ is *underdetermined* if there exist two solutions $\mu'$ and $\mu''$ of $DAE_G$ such that $\mu'_{|V_G \setminus \{y\}} = \mu''_{|V_G \setminus \{y\}}$ and $\mu'(y) \neq \mu''(y)$.

**Definition 3 (Electrical network semantics).** *The semantics of the network is defined by its associated $DAE_G$. We say that there exists a trajectory from a state $\mu$ to a state $\mu'$ if $\mu$ is a consistent initial value and there exists a continuously differentiable function $f : (0, t] \to V_G$ such that: $f(0) = \mu$, $f(t) = \mu'$, and for all $\delta \in (0, t]$, $\frac{df}{dt}(\delta)$ and $f(\delta)$ are a solution of $DAE_G$.*

**Structural analysis for electrical network.** Structural analysis for (non-switched) electrical networks is a standard technique used to determine if it is possible to *reformulate* the DAE into a system of *Ordinary Differential Equations* (ODEs). In the following, we will reuse established results from structural analysis. We use the standard definition of loops and cutset for a graph $G$. A sequence $n_0, e_0, \ldots, n_{k+1} \in N \times (E \times N)^k$ of nodes and edges is a *loop* if there exists a path from $n_0$ to $n_{k+1}$ (for $i \in [0, k]$, either $\eta(e_i) = (n_i, n_{i+1})$ or $\eta(e_i) = (n_{i+1}, n_i)$), and all the nodes are different, apart from $n_0$ and $n_{k+1}$ (for $i \in [0, k]$, $n_i \neq n_{i+1}$ and $n_0 = n_{k+1}$). The definition of loop ignores the edges orientation. We use the standard notion of subgraph, connected graph and connected component of a graph. If $G = \langle N, E, \eta \rangle$ is a connected graph, $K \subseteq E$ is a *cutset* of $G$ if removing $K$ from $E$ results in a disconnected graph, and $K$ is minimal (i.e. removing a proper subset of $K$ does not disconnect $G$). A loop is a V-loop (resp. VC-loop) if the only components on the edges are voltage sources (resp. voltage sources and capacitors). A cutset is an I-cutset (resp. IL-cutset) if the only components in the cutset are current sources (resp. current sources and inductors).

**Theorem 1 (Existence of an ODE reformulation (Theorem 6.3 from [24])).** *Given a connected electrical network G, the network has neither VC-loops nor IL-cutsets if and only if its associated $DAE_G$ can be reformulated into the ODE model:*

$$\dot{\boldsymbol{X}}_{\boldsymbol{G}} = A\boldsymbol{X}_{\boldsymbol{G}} + B\boldsymbol{U}_{\boldsymbol{G}} \qquad\qquad \boldsymbol{Y}_{\boldsymbol{G}} = C\boldsymbol{X}_{\boldsymbol{G}} + D\boldsymbol{U}_{\boldsymbol{G}} \tag{1}$$

*where $A \in \mathbb{R}^{|X_G|\times|X_G|}$, $B \in \mathbb{R}^{|X_G|\times|U_G|}$, $C \in \mathbb{R}^{|Y_G|\times|X_G|}$, $D \in \mathbb{R}^{|Y_G|\times|U_G|}$.*

The goal of the reformulation is to get the ODE, instead of a DAE, which are more amenable for simulation and verification.

The reformulation of $DAE_G$ as an ODE can be performed applying the *Superposition Theorem* [26]. The theorem tells that the response (the voltage and the current) of a component of a linear circuit is equal to the sum of the responses caused by each source acting alone (with all the other sources off). Turning on/off a voltage source means setting its voltage to 1/0 (the value for *on* must be different from 0), while turning on/off a current source means setting its current to 1/0. Capacitors and inductors are considered sources (of voltage and current respectively). The reformulation works by determining the contribution of each source (including inductors and capacitors) on the response (current or voltage) of each other component. Formally, for a component $e_i$ with a reformulated variable $w$, the reformulation works determining the coefficients $a_{w,z}$ such that:

$$w = \textstyle\sum_{z\in(X_G\cup U_G)} a_{w,z} z \tag{2}$$

where a coefficient $a_{w,z} \in \mathbb{R}$ represents the effect of the source variable $z$ on the reformulated variable $w$. $a_{w,z}$ is obtained considering only the effect of $z$, while disregarding the effects of the other sources. In practice, $a_{w,z}$ is the assignment to the variable $w$ in the system $DAE_G$ constrained by adding the constraints $z = 1$ and $l = 0$, for all the $l \in X_G \cup U_G \setminus \{z\}$.

**Switched Electrical Networks.** A *switch* $e_i$ is a component with two discrete states, *open* and *closed*. The state of the switch is represented with the Boolean variable $m_i$ (i.e. $m_i$ is true iff the switch is open). Let $M_i := \{m_i\}$ be the set of discrete variables, $C_i := X_i \cup U_i \cup Y_i$ the set of continuous variables and $V_i = M_i \cup C_i$ the set of all the variables of a switch. The constitutive relation of a switch is $\psi_i := \begin{cases} i_i = 0 & \text{if } m_i \\ v_i = 0 & \text{otherwise} \end{cases}$ (i.e. the switch disconnects or connects its terminals when it is open or closed). The switching behavior is defined by an invariant and a guard condition, $invar_i : 2^{M_i} \to \phi(C_i)$ and $guard_i : 2^{M_i} \to \phi(C_i)$. $invar_i$ defines the invariant condition of the switch that must hold in each discrete state, while $guard_i$ defines the condition that must hold in a discrete state to allow the transition to the other state.

**Definition 4 (Switched Electrical Network).** *A* switched electrical network $G = \langle N, E, \eta \rangle$ *is an electrical network where E may include also switches.*

We extend the set of variables defined for a component to the switched network in the obvious way. Also, let $E_m \subseteq E$ be the subset of all the switches components in $E$. We refer to each possible (complete) assignment $\mu$ to the discrete variables $M_G$ as a *discrete configuration* of the network, and we denote with $2^{M_G}$ the set of all the possible discrete configurations. Notice that, every different discrete configuration of the switches induces a (non-switched) electrical network. In the following, given a discrete configuration $\mu$, we refer to $DAE_G(\mu)$ as the DAE associated to the (non-switched) electrical network induced by $\mu$.

**Definition 5 (Valid switched electrical network.).** *We say that a switched electrical network G is* valid*, if, for all possible discrete configurations $\mu \in 2^{M_G}$, $DAE_G(\mu)$ can be reformulated into an ODE.*

In other words, a switched electrical network $G$ is *valid*, if, for all possible discrete configurations $\mu \in 2^{M_G}$:

(i) $DAE_G(\mu)$ has neither VC-loops nor IL-cutsets.
(ii) All the output variables $Y_G$ in $DAE_G(\mu)$ are not underdetermined.

**Definition 6 (Valid switched electrical network semantics.).** *We define the semantics of a valid switched electrical network $G = \langle N, E, \eta \rangle$ as the hybrid automaton $H_G = \langle D, R, Init, Invar, Trans, Flow \rangle$ where 1) $D := M_G$; 2) $R := C_G$; 3) $Init(D, R) := True$; 4) $Invar(D, R) := \bigwedge_{e_i \in E_m} (m_i \rightarrow invar_i(\{m_i\})) \wedge (\neg m_i \rightarrow invar_i(\emptyset))$; 5) $Trans(D, R, D', R') := (\bigvee_{e_i \in E_m} (m_i \wedge \neg m'_i \wedge guard_i(\{m_i\})) \vee (\neg m_i \wedge m'_i \wedge guard_i(\emptyset)) \wedge (\bigwedge_{x \in X_G} x' = x)$; 6) $Flow(D, \dot{R}, R) := DAE_G$;*

Notice that the flow conditions of the automaton that defines the semantic of the network still define a Differential-Algebraic Equation (DAE) and not an Ordinary Differential Equation (ODE).

## 3 Problem definition

In this paper, we address the following problems.

**Definition 7 (Network validation problem).** *The network validation problem consists of determining if a switched electrical network is valid. Additionally, if it is not the case, the problem also consists of finding the set of the discrete configurations that are not valid.*

A valid network can be encoded into a symbolic hybrid automaton where $Flow$ defines a system of ODEs for each configuration.

The hybrid automaton $H_G$ that defines the semantics of the network $G$ (see Definition 6) is a concise representation of the network. However, no model checking tools are able to analyze this kind of input (the combined symbolic representation and DAE). Thus, the problem that must be solved to enable the verification of a switched electrical network is the reformulation of the electrical switched network $G$ into a hybrid automaton with an ODE dynamics. Note that this problem extends the reformulation problem in Theorem 1 from a single DAE to a set of DAEs, one for each discrete configuration in $2^{M_G}$.

**Definition 8 (Hybrid Automata reformulation).** *Given a valid switched electrical network G, the reformulation problem consists of encoding G into a symbolic hybrid automaton with ODE dynamics.*

## 4 Network Validation

We show how to reduce the validation conditions to a series of SMT checks.

*SMT encoding.* Given a switched network $G = \langle N, E, \eta \rangle$, we encode the Differential-Algebraic Equation $DAE_G$ defined by the network as a quantifier free-formula in LRA. This formula will be used both for the validation and the reformulation steps.

The encoding formula predicates over the same variables of the network. We reuse the same notation for the different sets of variables used for the network $G$. In the encoding, we interpret each variable in $M_G$ as a Boolean variable and each variable in $X_G \cup U_G \cup Y_G$ as a Real variable. The encoding also predicates over the first-order derivatives of $X_G$, $\dot{X_G}$. We interpret each variable in $\dot{X_G}$ as a Real (the semantics should be clear from the context). The main reason is that both the validation and the reformulation just consider the algebraic relations defined by the equations, and not how the variables change as a function of time.

The formula $\psi_{DAE_G}$ connects the constitutive relation and voltage equation for each component $e_i$ through the $KCL$ and $KVL$ conditions:

$$\psi_{DAE_G} := \bigwedge_{e_i \in E \setminus E_{sources}} (\psi_i) \wedge \bigwedge_{e_i \in E} (v_i = v_i^+ - v_i^-) \wedge \psi_{KCL} \wedge \psi_{KVL}$$

$$\psi_{KCL} := \bigwedge_{n \in N} ( \sum_{e_i \in E_n^{in}} i_i - \sum_{e_i \in E_n^{out}} i_i = 0) \quad \psi_{KVL} := \bigwedge_{n \in N} ( \bigwedge_{p_1 \in P_n} ( \bigwedge_{p_2 \in P_n} p_1 = p_2))$$

*Existence of VC-loops or IL-cutsets.* As stated in Theorem 1, the DAE of a single configuration can be reformulated into an ODE if the network $G$ does not have any *VC-loops* or any *IL-cutsets* and if the network is connected. We encode these conditions in the following formulas.

$$\begin{aligned} val_z :=& \exists C_G, \dot{X_G}.(\psi_{DAE_G} \wedge z = 1 \wedge \textstyle\bigwedge_{l \in X_G \cup U_G \setminus \{z\}} l = 0) \\ val :=& \textstyle\bigwedge_{z \in X_G \cup U_G} val_z \end{aligned}$$

The formula $val_z$ sets to 1 the state or input variable $z$ of an active component (i.e. voltage sources, current sources, capacitors and inductors), while it keeps all the other state and input variables to 0. If $val_z$ is unsatisfiable for some discrete configuration in $2^{M_G}$, we have either a *VC-loop* or an *IL-cutset* involving $z$. This is due to the $KVL$ and the $KCL$ conditions. The first ensures that the sum of the voltages in a loop must be equal to 0. The latter ensures that the sum of the currents on the components in a cutset must be 0. For example, consider a configuration $\mu$ with a *VC-loop* that contains the voltage source $e_i$. The sum of the $KVL$ equations for the loop only contains variables from $X_G$ and $U_G$, and in particular the input variable $v_i$ of $e_i$. In the formula $val_{v_i}$ we have that $v_i = 1$, while all the other state and input variables are equal to zero. Thus, the $KVL$ equation of the loop reduces to $1 = 0$, and hence $val_{v_i}$ is unsatisfiable for $\mu$. An analogous reasoning can be done for an *IL-cutset* and the $KCL$ conditions.

**Lemma 1.** *The formula $val_{v_i}$ (resp. $val_{i_i}$) is satisfiable for all configurations $\mu \in 2^{M_G}$ if and only if the switched electrical network G does not have any VC-loops (resp. IL-cutsets) involving $v_i$ (resp. $i_i$).*

For lack of space, we provide the proofs in the Appendix C. As a corollary of Lemma 1, we have that the formula $val$ represents the set of all the configurations that do not have any *VC-loop* or *IL-cutset*. By Theorem 1, each configuration of the network admits a reformulation if there are no *VC-loops* or *IL-cutsets* and the network is connected.

*Existence of underdetermined output variables.* In a switched network, a configuration on a switch may induce a topology of the network that is not connected, but is formed by several connected components (of the graph of the network). The Theorem 1 can still be applied on each discrete configuration and on each connected component. In fact, for a network with neither *VC-loops* nor *IL-cutsets*, the theorem still guarantees the existence of the reformulation in terms of the state and input variables for each connected component of the graph containing at least an active component. We encode a sufficient condition for the connectedness of the network in the following formula:

$$und := \exists C_G, \dot{X}_G.(\psi_{DAE_G} \wedge \bigwedge_{z \in X_G \cup U_G}(z = 0) \wedge \bigvee_{y \in Y_G}(y \neq 0)) \qquad (3)$$

We consider the fact that all the output variables of a graph component are uniquely determined (i.e. are not underdetermined) by the input and state variables contained in such component if and only if the component is connected and does not show *degenerate* configurations such as *VC-loops* or *IL-cutsets*. The formula encodes that there exists a $y \in Y_G$ that can have a value different from 0 when all the input and state variables are 0. If the formula is satisfiable for some configuration $\mu$, then $y$ is underdetermined in that configuration.

**Lemma 2.** *The formula $und$ is satisfiable for some configuration $\mu \in 2^{M_G}$ if and only if there exists a variable $y \in Y_G$ that is underdetermined.*

As a corollary of Lemma 2, we have that $und$ represents the set of all the configurations that contain some underdetermined variable.

## 5 Network reformulation to Hybrid Automaton

### 5.1 Reformulation algorithm

Given a network $G = \langle N, E, \eta \rangle$, $H^r_G = \langle D^r, R^r, Init^r, Invar^r, Trans^r, Flow^r \rangle$ is the reformulated hybrid automaton. $H^r_G$ is defined as the hybrid automaton $H_G$ in the

Definition 6, except for $Invar^r$ and $Flow^r$. The invariant condition $Invar^r$ is given by $Invar^r := Invar \wedge Invar_Y^{ref}$, where $Invar$ is the invariant condition of $H_G$, and $Invar_Y^{ref}$ represents the reformulation of the output variables $Y_G$ (see Equation 2). $Flow^r$ represents the ODE dynamics in terms of $\dot{X}_G$, $X_G$, and $U_G$. The goal of the reformulation process is to synthesize both the $Flow^r$ and $Invar_Y^{ref}$ formulas.

In the following algorithms, we use a standard stack-based interface of an SMT solver (*push*, *assert*, *isSat*, *pop*, *reset* primitives). This allows us, after asserting a formula $\gamma$, to set a backtrack point (*push*), assert another formula $\beta$ (*assert*), check the satisfiability of the conjunction of the asserted formulas (*isSat*), and restore the state of the solver (i.e. asserted formulas and learned clauses) at the backtrack point (*pop*). This way, the satisfiability problem is solved keeping several learned clauses. Additionally, we assume to have the primitive *getModel*, to get a complete satisfying assignment to the free variables of the formula in the stack, and *quantify*, to eliminate the quantifiers present in the formula.

We describe a symbolic approach that groups together the discrete configurations that share the same ODE system. The algorithm REFORMULATE in Figure 3 reformulates only a subset of variables $W \subseteq \dot{X}_G \cup Y_G$. The algorithm can be used to reformulate all the dotted and output variables of the system by setting $W = \dot{X}_G \cup Y_G$. However, we will show how the modularity of REFORMULATE can be used to obtain different, and usually coarser, partitionings of the discrete configurations.

REFORMULATE takes as input the encoding of the network $\psi_{DAE_G}$, the sets of state and input variables $X_G$, $U_G$, and a set of variables $W$ to be reformulated. The main loop (line 3) of the algorithm enumerates all the discrete configurations of the network. Initially, the solver picks a random discrete configuration $\mu$ (line 4), and then symbolically applies the superposition theorem (on the network induced by $\mu$) calling the function GETCOEFFICIENTS (line 5). The output of GETCOEFFICIENTS is a map of coefficients $F$: for a $w \in W$ and a $z \in X_G \cup U_G$, $F(w)(z) \in \mathbb{R}$ is the coefficient that was obtained by observing the effect of the source $z$ on the variable $w$. Then, at line 6, the function GETEQMODES computes the set of all the *equivalent* discrete configurations $\beta$. GETEQMODES guarantees that $\mu' \in \beta$ if and only if GETCOEFFICIENTS finds the same coefficients when called on $\mu$ and on $\mu'$ with the same parameters $\psi_{DAE_G}$, $X_G$, $U_G$ and $W$. Then, at line 7, the algorithm *blocks* all the discrete configurations represented in $\beta$; this is a key step in the algorithm that prunes a set of discrete configurations from the search, avoiding their explicit enumeration. Finally, from line 8 to the end of the loop, REFORMULATE constructs the flow and invariant conditions.

The functions GETCOEFFICIENTS in Figure 6 implements the reformulation by the superposition theorem. Each execution of the loop at line 3, computes the effect of a state and input variable on all the variables in $W$.

The function GETEQMODES, shown in Figure 5, computes the set of configurations equivalent to $\mu$ in terms of reformulation. For each state and input variable, the function re-encodes the superposition conditions (line 4) and additionally encodes the coefficient constraints for the current discrete configurations $\mu$ (line 6). Then, the formula $\beta$ (line 9) encodes all the discrete configurations that have exactly the same reformulation of $\mu$. We also consider an alternative implementation of GETEQMODES,

```
REFORMULATE ($\psi_{DAE_G}$, $X_G$, $U_G$, $W$):
1. $(Flow_W, Invar_W) := (True, True)$
# Tautology over the variables $M_G$
2. solver.assert($\bigwedge_{m \in M_G} m \vee \neg m$)
3. while solver.isSat():
      # Get a configuration where $Flow_W$ is not defined
4.    $\mu$ := solver.getModel()
      # Get the coefficients that contribute to each $w \in W$
5.    $F$ := GETCOEFFICIENTS ($\psi_{DAE_G}$, $X_G$, $U_G$, $W$, $\mu$)
      # Get the modes that have the same dynamic for $W$
6.    $\beta$ := GETEQMODES ($\psi_{DAE_G}$, $X_G$, $U_G$, $W$, $F$)
      # Block the equivalent modes
7.    solver.assert($\neg\beta$)
8.    $(ref_{\dot{X}_G}, ref_{Y_G})$ := GETREF ($X_G$, $U_G$, $W$, $F$)
9.    $(Flow_W, Invar_W) := (Flow_W \wedge (\beta \rightarrow ref_{\dot{X}_G}), Invar_W \wedge (\beta \rightarrow ref_{Y_G}))$
10. return $(Flow_W, Invar_W)$
```

**Fig. 3.** Reformulate a set of variables $W$.

GETEQMODESMODULAR, that computes the existential quantification independently for each single conjunct of the formula $\gamma$, instead of the whole formula $\gamma$.

## 5.2 All and single variables partitioning

The REFORMULATE algorithm allows us to reformulate sets of variables (i.e. subsets of $\dot{X}_G \cup Y_G$) instead of all the variables $\dot{X}_G \cup Y_G$. Thus, it allows us to obtain different kinds of partitioning of the discrete configurations and the ordinary differential equations. We define two reformulation algorithms that obtain different partitioning. The ALLREF algorithm, shown at the top of Figure 7, first reformulates all the controlled variables $X_G$. Thus, in this case we obtain sets of discrete configurations that share the same system of ODEs (a system of equations of the form $\dot{\boldsymbol{X}}_{\boldsymbol{G}} = A\boldsymbol{X}_{\boldsymbol{G}} + B\boldsymbol{U}_{\boldsymbol{G}}$). Then, ALLREF reformulates all the output variables $Y_G$ independently.

```
GETREF ($X_G$, $U_G$, $W$, $F$):
1. $(ref_{\dot{X}_G}, ref_{Y_G})$ := $(True, True)$
2. for each $w \in W$:
3.    $rhs_w$ := 0
4.    for each $z \in X_G \cup U_G$:
5.       $rhs_w := rhs_w + F(w)(z) * z$
6.    if $w \in \dot{X}_G$ :
7.       $ref_{\dot{X}_G} := ref_{\dot{X}_G} \wedge w = rhs_w$
8.    else:
9.       $ref_{Y_G} := ref_{Y_G} \wedge w = rhs_w$
10. return $(ref_{\dot{X}_G}, ref_{Y_G})$
```

**Fig. 4.** Construction of the reformulation formulas.

The other algorithm, SINGLEREF (shown in the bottom of Figure 7), instead reformulates all the variables independently.

As a further observation, in practice we do not need to reformulate all the output variables $Y_G$. In fact, we need to reformulate only those variables needed to define the dynamics of the system (e.g. they may be used in the invariant $invar_i$ and $guard_i$ conditions of a switch) or the variables that we want to observe.

```
GETEQMODES ($\psi_{DAE_G}, X_G, U_G, W, F$):
1. eqSolver.reset()
2. $\gamma := True$
3. for each $z \in X_G \cup U_G$:
4.     $sup_z := z = 1 \wedge \bigwedge_{l \in (X_G \cup U_G) \setminus \{z\}} l = 0$
5.     $\gamma_F := True$
6.     for each $w \in W$:
7.         $\gamma_F := \gamma_F \wedge w = F(w)(z)$
8.     $\gamma := \gamma \wedge \exists C_G, \dot{X}_G.(\psi_{DAE_G} \wedge \gamma_F \wedge sup_z)$
9. $\beta$ := eqSolver.quantify($\gamma$)
10. return $\beta$
```

**Fig. 5.** Find the discrete configurations with an equivalent dynamics for a set of variables $W$.

```
GETCOEFFICIENTS ($\psi_{DAE_G}, X_G, U_G, W, \mu$):
# F maps vars in W and $X_G \cup U_G$ to real values
1. $F : W \rightarrow (X_G \cup U_G) \rightarrow \mathbb{R}$
2. coeffSolver.assert($\psi_{DAE_G} \wedge \mu$)
3. for each $z \in X_G \cup U_G$:
4.     coeffSolver.push()
5.     $sup_z := z = 1 \wedge \bigwedge_{l \in (X_G \cup U_G) \setminus \{z\}} l = 0$
6.     coeffSolver.assert($sup_z$)
7.     $\mu'$ := coeffSolver.getModel()
8.     for each $w \in W$:
           # $\mu'(w)$ represents the effect of $z$ on $w$
9.         $F(w)(z) := \mu'(w)$
10.    coeffSolver.pop()
11. return $F$
```

**Fig. 6.** Computes the superposition coefficients for a set of variables $W$.

## 6 Related work

The solutions to the validation and reformulation problems for electrical networks (without switches) are well known [5, 24]. We differ from these works since we focus on networks with discrete switches, where the main issue is to cope with the exponential explosion in the number of discrete configurations. Then, we reuse several techniques from structural analysis, as the superposition principle [26], but we re-interpret them in a symbolic setting.

Other works consider also networks with discrete switches. Several approaches [18] do not consider ideal switches, but model the switch introducing parasitic resistances. A drawback of this approach is that it requires to determine a priori a set of parameters (e.g the resistance of the resistor); then, these parameters have the effect to change the dynamics of the systems, producing as a result an approximation of the intended behavior. Ideal switches have been mainly considered in context of simulation, for example in [17]. While the focus is often on non-linear dynamics, the problem solved in these works is to produce a single simulation of the network. In this context, they reformulate the DAE into an ODE every time the simulator performs a discrete switch. Thus, these works do not solve the validation and the reformulation problem, since they focus on a single execution of the system.

Several works focus on the translation from Stateflow/Simulink models to hybrid automata [1, 16, 19]. We point out that the Simulink modeling is based on a functional representation of a system where every block is seen as an unidirectional Input-Output function, thus it is not suitable for a component-based physical modeling that is intrinsically bidirectional. There are several works [29] on the formal verification of Analog-Mixed-Signal (AMS) circuits. Most works focus on non-switched circuits [9,10,15] and try to solve a reachability problem. They start from the network representation but they manually encode it as a hybrid automaton. A different approach is considered in [30], where a non-linear circuit is automatically abstracted and encoded using SMT. We re-

```
AllRef ($\psi_{DAE_G}, X_G, U_G$):
1.$(Flow^r, Invar^{ref}_Y)$ := Reformulate ($\psi_{DAE_G}, X_G, U_G, \dot{X}_G$)
2.for each $w \in Y_G$:
3.    $(Flow_w, Invar_w)$ := Reformulate ($\psi_{DAE_G}, X_G, U_G, \{w\}$)
4.    $(Flow^r, Invar^{ref}_Y)$ := $(Flow^r \wedge Flow_w, Invar^{ref}_Y \wedge Invar_w)$
5.return $(Flow^r, Invar^{ref}_Y)$

SingleRef ($\psi_{DAE_G}, X_G, U_G$):
1.$(Flow^r, Invar^{ref}_Y)$ := $(True, True)$
2.for each $w \in \dot{X}_G \cup Y_G$:
3.    $(Flow_w, Invar_w)$ := Reformulate ($\psi_{DAE_G}, X_G, U_G, \{w\}$)
4.    $(Flow^r, Invar^{ref}_Y)$ := $(Flow^r \wedge Flow_w, Invar^{ref}_Y \wedge Invar_w)$
5.return $(Flow^r, Invar^{ref}_Y)$
```

**Fig. 7.** Reformulation algorithms with different reformulation strategies.

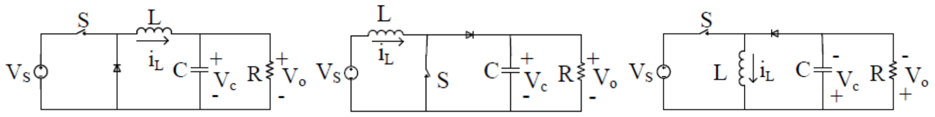
mark that none of these works solves the validation and reformulation problem for a switched electrical network.

Finally, our reformulation approach produces a symbolic hybrid automata model that can be analyzed by model checkers tools like HybridSAL [27] and HyComp [6], using relational abstraction [20, 28], or dReach [3, 14].

## 7 Experimental evaluation

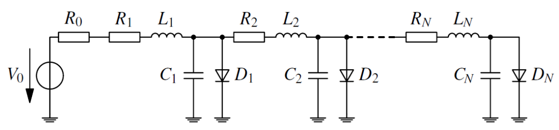
The approach was implemented in pySMT [11], a library for SMT formulae manipulation and solving, using MathSat5 [7] for Quantifier Elimination. We evaluated the effectiveness and the scalability of the symbolic approach in the validation and reformulation problems. The experimental evaluation was run on a 64 bit system with an Intel Xeon E3-1246 processor at 3.5 GHz and 16GB RAM. The tool and the benchmarks used in the experiments are available at `https://es.fbk.eu/people/sessa/attachment/FM2016/fm16.tar.bz2`

**Benchmarks.** We consider several classes of benchmarks. The following (Buck, Boost, Buck-Boost) *DC-DC converters* are taken from [21].



The *Switched RC Network* SRCN$_N$ is a scalable benchmark obtained from the circuit of Figure 2 by parameterizing the number of (up to 8) capacitive branches.

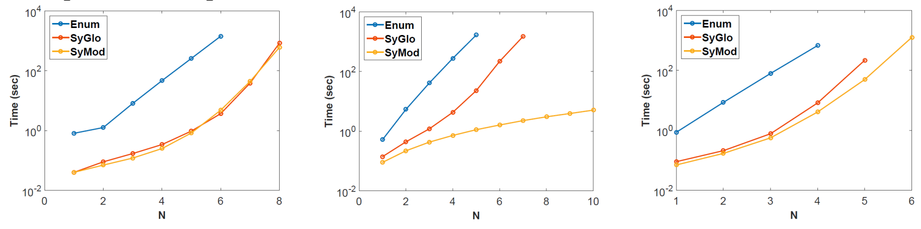
The *Non Linear Transmission Line (NLTL)* depicted below represents a well-known phenomenon (discretization of propagation) along a transmission line [15]. We parameterize the benchmark NLTL$_N$ on the number $N$ of (up to 10) pairs of stages.



The *Wheel Braking System* benchmarks follow the description in the SAE Standard AIR6110 [25] (see Figure 1). We consider the WBS$_N$ benchmarks, parameterized on the number of (up to 6) braking lines. The WBS consists of a pressure supply line made of a pump, an accumulator, pipelines and an isolation valve, connected to replicas of a braking line made of pipelines, distribution valves, fuses and brakes. Following the Electronic-Hydraulic analogy, the pump is modeled as an ideal constant voltage source, the pipes as resistors, the accumulator and the brakes as capacitors, the distribution valves and the fuses as ideal switches, and the isolation valve as a diode.

For each of the scalable benchmarks, the number of discrete configurations grows exponentially with the problem size, reaching a million of system configurations for the NLTL$_{10}$. Additional information are available in the Table 4 in Appendix D.

**Validation.** We first consider the results of the validation. For the scalable benchmarks, we report the comparison of three different strategies: the baseline *Enum* strategy, explicitly enumerates the system configurations and for each of them validates the induced DAE; the *SyGlo* and *SyMod* strategies apply quantifier elimination (QE) over two different SMT encoding of the validation problem. The former tries to minimize the number of QEs encoding the validation problem into a global SMT formula, the latter tries to reduce the complexity of the global QE decomposing the global encoding into a modular sequence of simpler formulas. From the results (SRCN, NLTL, WBS from the left)

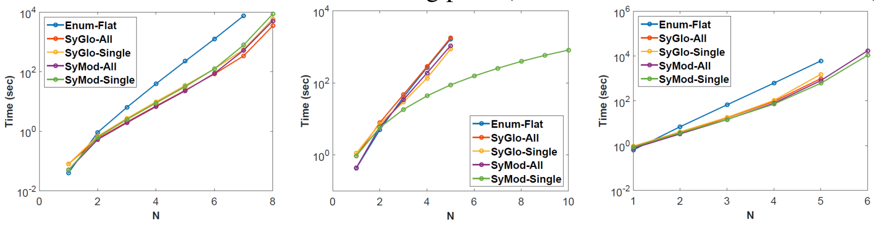


we see that the symbolic approaches outperform the enumerative approach at least of one order of magnitude in all the benchmarks. While the two symbolic approaches show similar performance on the SRCN, for the WBS and NLTL *SyMod* accomplishes the task while *SyGlo* times out. In general, *SyMod* performs much better than *SyGlo*.

The non-scalable benchmarks are validated all within one second, but provide interesting insights. Specifically, the models of the DC-DC converters result in four discrete configurations, given by the switch $S$ and diode $D$. The hybrid automata provided in [21] only contain the two discrete modes $S = open, D = closed$ and $S = closed, D = open$. In fact, the validation phase detects (for each converter) two non valid configurations, corresponding to $S = open, D = open$ and $S = closed, D = closed$, that induce an IL-cutset and a VC-loop, respectively. These two modes are exactly those excluded in the manual modeling phase leading to the hybrid automata provided in [21].

**Reformulation.** In the reformulation phase, for each model, we reformulate all the *derivative* variables and only the *output* variables contained in the invariant and guard formulas of the system components (e.g. the voltage and current of a diode).

Applying the reformulation phase to the DC-DC converters restricted to only the two valid configurations $S = open, D = closed$ and $S = closed, D = open$, we get two distinct ODEs whose coefficients agree with the dynamics of the converters provided in [21]. For lack of space, we refer the reader to the Figure 9 in the Appendix D for further details on the output of the converters reformulation.
For each scalable benchmark, the following plots (SRCN, NLTL, WBS from the left)



show the reformulation time for five different approaches that mix the *Enum* and *Symbolic* strategies with different *reformulation strategies*. *Enum-Flat* represents the naive approach that enumerates the system configurations and reformulates the derivative and output variables as a unique set of variables; the *SyGlo-All* and the *SyMod-All* approaches apply the ALLREF reformulation algorithm; the *SyGlo-Single* and the *SyMod-Single* approaches apply the SINGLEREF algorithm. In general, the two symbolic approaches outperform the enumerative approach and exhibit similar performance, with the exception of the *SyMod-Single* reformulation, that has significant advantage over the others in the NLTL benchmarks due to its favorable topology. The choice of the strategy (*All* vs *Single*) in the symbolic reformulation affects the amount of discovered equivalence classes, that is directly correlate with the reformulation time. Additional details are reported in Figures 10, 11, 12 and Figures 13, 14, 15 in the Appendix D.

## 8 Conclusion

In this paper we presented a novel, symbolic approach to the validation and reformulation problem of a switched electrical network. The method is able to analyze the validity conditions of the network, where the dynamics are expressed as Differential-Algebraic Equations, and to reformulate them in form of a (symbolically represented) Hybrid Automaton. The proposed approach scales much better than an naive approach based on the enumerative analysis of the individual configurations, and produces significantly more compact HA due to the clustering of the equivalent configurations.

In the future, we will explore, amongst other research directions, how the approach can be generalized to other physical domains (e.g. mechanical) where the conditions needed for the network validation are different, and to deal with partially underdetermined networks.

## References

1. Agrawal, A., Simon, G., Karsai, G.: Semantic translation of simulink/stateflow models to hybrid automata using graph transformations. Electronic Notes in Theoretical Computer Science 109, 43 – 56 (2004), http://www.sciencedirect.com/science/article/pii/S1571066104052089, proceedings of the Workshop on Graph Transformation and Visual Modelling Techniques (GT-VMT 2004)
2. Akers, A., Gassman, M., Smith, R.: Hydraulic Power System Analysis. Fluid Power and Control, CRC Press (2006), https://books.google.it/books?id=Uo9gpXeUoKAC
3. Bae, K., Kong, S., Gao, S.: SMT encoding of hybrid systems in dReal. In: Frehse, G., Althoff, M. (eds.) ARCH14-15. 1st and 2nd International Workshop on Applied veRification for Continuous and Hybrid Systems. EPiC Series in Computing, vol. 34, pp. 188–195. EasyChair (2015)
4. Barrett, C.W., Sebastiani, R., Seshia, S.A., Tinelli, C.: Satisfiability modulo theories. In: Handbook of Satisfiability, pp. 825–885 (2009), http://dx.doi.org/10.3233/978-1-58603-929-5-825
5. Benner, P.: Large-scale networks in engineering and life sciences. Springer, Birkhäuser Mathematics (2014)
6. Cimatti, A., Griggio, A., Mover, S., Tonetta, S.: HyCOMP: An SMT-based model checker for hybrid systems. In: TACAS. pp. 52–67 (2015), http://dx.doi.org/10.1007/978-3-662-46681-0_4
7. Cimatti, A., Griggio, A., Schaafsma, B.J., Sebastiani, R.: Tools and Algorithms for the Construction and Analysis of Systems: 19th International Conference, TACAS 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings, chap. The MathSAT5 SMT Solver, pp. 93–107. Springer Berlin Heidelberg, Berlin, Heidelberg (2013), http://dx.doi.org/10.1007/978-3-642-36742-7_7
8. Cimatti, A., Mover, S., Tonetta, S.: A quantifier-free SMT encoding of non-linear hybrid automata. In: FMCAD. pp. 187–195 (2012), http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6462573
9. Dang, T., Donzé, A., Maler, O.: Verification of analog and mixed-signal circuits using hybrid system techniques. In: Formal Methods in Computer-Aided Design, 5th International Conference, FMCAD 2004, Austin, Texas, USA, November 15-17, 2004, Proceedings. pp. 21–36 (2004), http://dx.doi.org/10.1007/978-3-540-30494-4_3
10. Frehse, G., Krogh, B.H., Rutenbar, R.A., Maler, O.: Time domain verification of oscillator circuit properties. Electr. Notes Theor. Comput. Sci. 153(3), 9–22 (2006), http://dx.doi.org/10.1016/j.entcs.2006.02.019
11. Gario, M., Micheli, A.: pysmt: a solver-agnostic library for fast prototyping of smt-based algorithms. In: SMT Workshop (2015)
12. Henzinger, T.A.: The theory of hybrid automata. In: Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996. pp. 278–292 (1996), http://dx.doi.org/10.1109/LICS.1996.561342
13. Janschek, K.: Mechatronic systems design: methods, models, concepts. Springer Science & Business Media (2011)
14. Kong, S., Gao, S., Chen, W., Clarke, E.M.: dReach: $\delta$-reachability analysis for hybrid systems. In: TACAS. pp. 200–205 (2015), http://dx.doi.org/10.1007/978-3-662-46681-0_15
15. Lee, H.L., Althoff, M., Hoelldampf, S., Olbrich, M., Barke, E.: Automated generation of hybrid system models for reachability analysis of nonlinear analog circuits. In: The 20th Asia

and South Pacific Design Automation Conference, ASP-DAC 2015, Chiba, Japan, January 19-22, 2015. pp. 725–730 (2015), http://dx.doi.org/10.1109/ASPDAC.2015.7059096

16. Manamcheri, K., Mitra, S., Bak, S., Caccamo, M.: A step towards verification and synthesis from simulink/stateflow models. In: Proceedings of the 14th ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2011, Chicago, IL, USA, April 12-14, 2011. pp. 317–318 (2011), http://doi.acm.org/10.1145/1967701.1967749
17. Massarini, A., Reggiani, U., Kazimierczuk, M.K.: Analysis of networks with ideal switches by state equations. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications 44(8), 692–697 (Aug 1997)
18. Mathworks, T.: Simscape power systems, http://it.mathworks.com/help/physmod/sps/index.html
19. Minopoli, S., Frehse, G.: SL2SX translator: From simulink to spaceex models. In: Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control, HSCC 2016, Vienna, Austria, April 12-14, 2016. pp. 93–98 (2016), http://doi.acm.org/10.1145/2883817.2883826
20. Mover, S., Cimatti, A., Tiwari, A., Tonetta, S.: Time-aware relational abstractions for hybrid systems. In: EMSOFT. pp. 14:1–14:10 (2013), http://dx.doi.org/10.1109/EMSOFT.2013.6658592
21. Nguyen, L.V., Johnson, T.T.: Benchmark: DC-to-DC switched-mode power converters (buck converters, boost converters, and buck-boost converters). In: Frehse, G., Althoff, M. (eds.) ARCH14-15. 1st and 2nd International Workshop on Applied veRification for Continuous and Hybrid Systems. EPiC Series in Computing, vol. 34, pp. 19–24. EasyChair (2015)
22. Nuzzo, P., Xu, M., Ozay, N., Finn, J.B., Sangiovanni-Vincentelli, A., Murray, R., Donze, A., Seshia, S.: A contract-based methodology for aircraft electric power system design. IEEE Access (November 2014), http://icyphy.org/pubs/35.html
23. Riaza, R.: Differential-algebraic systems: analytical aspects and circuit applications. World Scientific (2008)
24. Riaza, R.: Differential-algebraic systems analytical aspects and circuit applications. World Scientific, Singapore, SG (2008)
25. SAE International: AIR 6110 - Contiguous Aircraft/System Development Process Example (2011)
26. Skaar, D.L.: Using the superposition method to formulate the state variable matrix for linear networks. IEEE Transactions on Education 44(4), 311–314 (Nov 2001)
27. Tiwari, A.: HybridSAL Relational Abstracter. In: CAV. pp. 725–731 (2012)
28. Tiwari, A.: Time-aware abstractions in HybridSal. In: CAV. pp. 504–510 (2015), http://dx.doi.org/10.1007/978-3-319-21690-4_34
29. Zaki, M.H., Tahar, S., Bois, G.: Formal verification of analog and mixed signal designs: Survey and comparison. In: 2006 IEEE North-East Workshop on Circuits and Systems. pp. 281–284 (June 2006)
30. Zhang, Y., Sankaranarayanan, S., Somenzi, F.: Piecewise linear modeling of nonlinear devices for formal verification of analog circuits. In: FMCAD. pp. 196–203 (2012), http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6462574

## A Motivating example

Figure 2 shows an *electrical switched network* that contains a voltage source ($V_s$), resistors ($R_0, R_1, R_2$), switches ($S_1, S_2$), fuses ($F_1, F_2$) and capacitors ($C_1, C_2$).

The network contains $4$ different switching elements: the switches and the fuses. Both kind of components may be closed or open, respectively connecting or disconnecting a branch of the circuit. While a switch opens or closes non-deterministically [4], the fuse opens or closes depending on the current that flows through it (i.e. the switching is autonomous in this case). Since there are $4$ different switching elements, which can be either open or closed, the network has in total $2^4$ discrete configurations. In turn, each configuration induces a different topology of the network. For example, in the network of Fig. 2, when all the switching elements are closed we have the circuit where the branch with $R_1$ and $C_1$ is connected in parallel with the branch with $R_2$ and $C_2$, and both of them are connected in series with the generator $V_s$. We obtain a new configuration by just opening $S_2$, which disconnects $R_2$ and $C_2$.

The final goal of our approach is to validate the network and to obtain a hybrid automaton that is equivalent to the network, and that can be further used to verify functional properties of the network.

Obtaining a hybrid automaton is a "simple" problem. The naive approach consists of enumerating all the discrete configurations and, for each configuration, either obtain an Ordinary Differential Equation (*reformulation step*) or report that the configuration is not valid (*validation step*). The final result is a hybrid automaton that has **a location for each discrete configuration** of the network. For example, consider the configuration where $S_1$ and $F_1$ are closed while $S_2$ and $F_2$ are open. The whole procedure consists of the following steps.

1. Obtain a Differential-Algebraic Equation (Figure 8) that describes the behavior of the system (there exist several methods, for example the *Sparse Tableau Formulation* or *Modified Nodal Analysis*).

$$
\begin{array}{llll}
V_{R_0} = R_0 I_{R_o} & V_{R_0} = V_{R_0}^{+}\text{-}V_{R_0}^{-} & V_S^{-} = 0 & I_S - I_{R_0} = 0 \\
V_{R_1} = R_1 I_{R_1} & V_{R_1} = V_{R_1}^{+}\text{-}V_{R_1}^{-} & V_S^{-} = V_{C_1}^{-} & I_{R_0} - I_{R_1} = 0 \\
V_{R_2} = R_2 I_{R_2} & V_{R_2} = V_{R_2}^{+}\ V_{R_2}^{-} & V_S^{-} = V_{C_2}^{-} & I_{R_1} - I_{C_1} = 0 \\
C_1 \dot{V}_{C_1} = I_{C_1} & V_{C_1} = V_{C_1}^{+}\text{-}V_{C_1}^{-} & V_S^{+} = V_{R_0}^{+} & I_{R_2} = 0 \\
C_2 \dot{V}_{C_2} = I_{C_2} & V_{C_2} = V_{C_2}^{+}\text{-}V_{C_2}^{-} & V_{R_0}^{-} = V_{R_1}^{+} & I_{R_2} - I_{C_2} = 0 \\
 & V_S = V_S^{+}\text{-}V_S^{-} & V_{R_1}^{-} = V_{C_1}^{+} & I_{C_1} + I_{C_2} - I_S = 0 \\
 & & V_{R_2}^{-} = V_{C_2}^{+} &
\end{array}
$$

**Fig. 8.** DAE

2. Validate the network. This step reduces to verify some topological properties on the network graph, namely, the absence of *VC-loops* and *IL-cutsets* and the connectedness. (see Sec. 3).

[4] In this paper we do not consider external controllers that interact with the circuit.

For instance, consider the circuit of Figure 2 extended with a new switch $S_3$ connected to the left-hand side terminals of the capacitors $C_1$ and $C_2$. Such new circuit has $2^5$ configurations. When $S_3$ is open we get the $2^4$ configurations of the original circuit, but when $S_3$ is closed we get $2^4$ new configurations that contain the *VC-loop* of the two capacitors $C_1$ and $C_2$.
In this case the hybrid automaton cannot be built, and the designer must be informed, reporting that those configurations are *degenerate*.

3. Rewrite the DAE as an ODE by algebraic manipulations. In our example, if $R_0 = R_1 = R_2 = 1$ and $C_1 = C_2 = 2$, the corresponding ODE is:

$$\dot{v}_{c_1} = -\frac{1}{4}v_{c_1} + 0v_{c_2} + \frac{1}{4}v_S \tag{4}$$

$$\dot{v}_{c_2} = 0v_{c_1} + 0v_{c_2} + 0v_S \tag{5}$$

The main issue with switched systems is that the number of discrete configurations of the network is exponential in the number of switching elements. Thus, explicitly enumerating these configurations is not feasible. To overcome the problem, both for the validation and reformulation step, we use a symbolic representation of the network and we use a Satisfiability Modulo Theory (SMT) solver to reason over it.

The validation step is reduced to check the satisfiability of two existentially quantified formulas, which implicitly encodes the DAEs of the exponential number of discrete configurations.

The reformulation step returns a *symbolic* hybrid automaton, where all the discrete configurations that have the same ODE are grouped together. For example, the hybrid automaton of the example in Fig. 2 has $4$ "symbolic locations", instead of $2^4$. The approach randomly pick a fresh discrete configuration, computes its ODE reformulation and then finds and group together all the other *equivalent* configurations, which have the same ODE. The Table 2 shows such partitioning for the circuit of Figure 2. For the sake of conciseness, we represent the sixteen system configuration as a decimal number, from 0 to 15. Every number corresponds to the decimal conversion of the binary word $\langle S_1, F_1, S_2, F_2 \rangle$ where every element takes value 0 if closed, and 1 if open.

| $ODE_i$ for $\dot{v}_{c_1}, \dot{v}_{c_2}$ | $ODE_1$ | $ODE_2$ | $ODE_3$ | $ODE_4$ |
|---|---|---|---|---|
| Configuration eq. class | {0} | {1,2,3} | {4,8,12} | {5,6,7,9,10,11,13,14,15} |

**Table 2.** Partitioning of the configurations by equivalence on the global ODE.

Moreover, we further explore how we can obtain different partitionings of the discrete configurations of the system. The idea is to group together all the discrete configurations that define the same ODE for a single variable, instead of grouping the configurations that have the same system of ODEs (i.e. we look at a single variable and not at the whole set of controlled variables of the system). For example, consider again the circuit of Figure 2. The partitioning that considers the single variables reformulation is shown in Table 3.

| $ODE_i^1$ for $\dot{v}_{c_1}$ | $ODE_1^1$ | $ODE_2^1$ | $ODE_3^1$ |
|---|---|---|---|
| Configuration eq. class | {0} | {1,2,3} | {4,5,6,7,8,9,10,11,12,13,14,15} |
| $ODE_i^2$ for $\dot{v}_{c_2}$ | $ODE_1^2$ | $ODE_2^2$ | $ODE_3^2$ |
| Configuration eq. class | {0} | {4,8,12} | {1,2,3,5,6,7,9,10,11,13,14,15} |

**Table 3.** Partitioning of the configurations by equivalence on the single ODE variable.

In practice, in the partitioning we can consider different levels of granularity: we can group together configurations that have the same ODE for an arbitrary subset of the variables of the system. By exploring different partitioning, our conjecture is of being able to obtain larger sets of equivalent discrete configurations, as shown in Tables 2, 3, leading to more concise hybrid automata through a reduces number of reformulations. The trade-off is not immediate, since we need to perform a reformulation for each variable of the system.

## B Hybrid automaton semantics

A *state* of a hybrid automaton $H = \langle D, R, Init, Invar, Trans, Flow\rangle$ is an assignment $s$ to the variables $D \cup R$.

**Definition 9 (Run of the hybrid automaton).** *A sequence of states* $s_0 \xrightarrow{\delta_1} s_1 \xrightarrow{\delta_2} \ldots \xrightarrow{\delta_k} s_k$ *is a* path *of the hybrid automaton* $H$ *if:*

- $s_0 \models Init$ *and for* $0 < i \leq k$, $s_i$ *is a state of* $H$*;*
- *for* $1 \leq i \leq k$, $\delta_j \in \mathbb{R} \cup \{d\}$ *and* $s_{i-1} \xrightarrow{\delta_i} s_i$ *we have that either: 1) Discrete transition:* $\delta_i = d$, $\langle s_{i-1}, \delta_i, s_i\rangle \models Trans$, $s_{i-i} \models Invar$ *and* $s_i \models Invar$. *2) Continuous transition:* $\delta_i \in \mathbb{R}, \delta_i > 0$, $s_{i-1|D} = s_{i|D}$, *and there exists a continuous differentiable function* $f : [0, \delta_i] \rightarrow \mathbb{R}^{|R|}$ *such that:* $f(0) = s_{i-1|R}$ *and* $f(\delta_i) = s_{i|R}$, $s_{i-1} \models Invar$, $s_i \models Invar$, $\forall \epsilon \in [0, \delta_i]$, $\langle s_{i-1|D}, f(\epsilon), \dot{f}(\epsilon)\rangle \models Flow$, $\forall \epsilon \in [0, \delta_i], \langle s_{i-1|D}, f(\epsilon)\rangle \models Invar$.

## C Proofs

**Lemma 1.** *The formula* $val_{v_i}$ *(resp.* $val_{i_i}$*) is satisfiable for all configurations* $\mu \in 2^{M_G}$ *if and only if the switched electrical network G does not have any VC-loops (resp. IL-cutsets) involving* $v_i$ *(resp.* $i_i$*).*

*Proof.* In the following, we prove the lemma in the case of the variable $v_i$ of the component $e_i$, while the proof in the case of $i_i$ is analogous, considering as equation the sum of the current on all the elements of a cutset, instead of the sum of all the voltage in a loop.

($\Rightarrow$) If $val_{v_i}$ is satisfiable for all configurations, then $G$ does not have any *VC-loops*.

By absurd, suppose that for a discrete configuration $\mu \in 2^{M_G}$ of $G$ there exists a *VC-loop* on the component $e_i$. Then, there exists a loop $n_0, e_0, \ldots, n_{k+1} \in N \times$

$(E \times N)^k$, where $e_j = e_i$, for $j \in [0, k]$, and we have the following $KVL$ condition $\sum_{e \in \{e_0,\ldots,e_k\}} v_e = 0$. Notice that this condition is encoded in the $\psi_{DAE_G}$.

Then the formula $val_{v_i}$ is not satisfiable for $\mu$ ($v_i$ is $1$ and the other $e$-s are $0$), deriving a contradiction.

($\Leftarrow$) If $G$ does not have any *VC-loops*, then $val_{v_i}$ is satisfiable for all configurations.

By absurd, suppose that $val_{v_i}$ is unsatisfiable for $\mu$. By construction, we have that the DAE for each discrete configuration is a homogeneous system that has a solution if all the continuous variables $C_G$ are set to 0 (in particular $\psi_{DAE_G} \wedge \bigwedge_{z \in X_G \cup U_G} z = 0$ is satisfiable). Thus, we have that $val_{v_i}$ is unsatisfiable because of $v_i = 1 \wedge \bigwedge_{l \in X_G \cup U_G \setminus \{v_i\}} l = 0$ and the $KVL$ equations that relate the set of voltage variables of the network. Then, there exists a *VC-loop*.

**Lemma 2.** *The formula $und$ is satisfiable for some configuration $\mu \in 2^{M_G}$ if and only if there exists a variable $y \in Y_G$ that is underdetermined.*

*Proof.* ($\Rightarrow$) If $und$ is satisfiable for some configuration, then there exists a variable $y \in Y_G$ that is underdetermined.

If $und$ is satisfiable for some configuration, then we have two assignments to the continuous variables of the network that differ for the value of the variable $y$. This follows from the assumption that the DAE is homogeneous.

($\Leftarrow$) If there exists a variable $y \in Y_G$ that is underdetermined in $\mu$, then $und$ is satisfiable for $\mu$.

There exist two configurations of the network, $\mu'$ and $\mu''$, that differ only for the value of $y$ (since $y$ is underdetermined). Since the DAE is homogeneous, at least one of the two configurations satisfies $y \neq 0$. This makes the term $\bigvee_{y \in Y_G} (y \neq 0))$ true.

The other term $\psi_{DAE_G} \wedge \bigwedge_{z \in X_G \cup U_G} (z = 0)$ is trivially satisfied due to the homogeneity of the DAE.

Thus, $und$ is satisfiable.

## D Additional Experimental Results

### D.1 DC-DC converter reformulation

In Figure 9 we show the ODEs resulting from the reformulation of the DC-DC converters restricted to the valid configurations. The benchmarks have been instantiated with the following parameters: $R = 3$, $C = 2$, and $L = 5$.

### D.2 Scalable benchmarks

In this section we present additional experimental results for the scalable benchmarks. Table 4 show the size of the benchmarks in terms of *system configurations* and *reformulation variables*.

Figures 10, 11 12 show the total amount of equivalence classes discovered by the reformulation strategies.

Figures 13, 14 15 show the size of the largest mode partition discovered by the different reformulation strategies. Noteworthy, in the case of the NLTL benchmark reformulated with the *Single* granularity, the largest partition has size 2 independently from the problem size.

|  | $S = open, D = closed$ | $S = closed, D = open$ |
|---|---|---|
| *Buck converter* | $\dot{i}_L = 0i_L - \frac{1}{5}v_C + 0v_S$<br>$\dot{v}_C = \frac{1}{2}i_L - \frac{1}{6}v_C + 0v_S$ | $\dot{i}_L = 0i_L - \frac{1}{5}v_C + \frac{1}{5}v_S$<br>$\dot{v}_C = \frac{1}{2}i_L - \frac{1}{6}v_C + 0v_S$ |
| *Boost converter* | $\dot{i}_L = 0i_L - \frac{1}{5}v_C + \frac{1}{5}v_S$<br>$\dot{v}_C = \frac{1}{2}i_L - \frac{1}{6}v_C + 0v_S$ | $\dot{i}_L = 0i_L + 0v_C + \frac{1}{5}v_S$<br>$\dot{v}_C = 0i_L - \frac{1}{6}v_C + 0v_S$ |
| *Buck − Boost converter* | $\dot{i}_L = 0i_L - \frac{1}{5}v_C + 0v_S$<br>$\dot{v}_C = \frac{1}{2}i_L - \frac{1}{6}v_C + 0v_S$ | $\dot{i}_L = 0i_L + 0v_C + \frac{1}{5}v_S$<br>$\dot{v}_C = 0i_L - \frac{1}{6}v_C + 0v_S$ |

**Fig. 9.** Reformulation of the DC-DC converters restricted to the valid configurations.

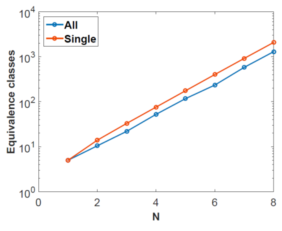


**Fig. 10.** SRCN. Cumulative mode partitioning.

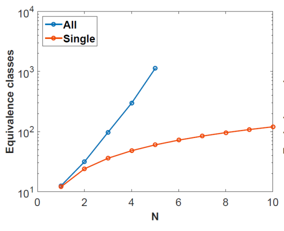


**Fig. 11.** NLTL. Cumulative mode partitioning.

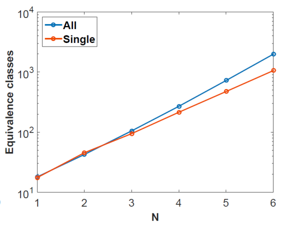


**Fig. 12.** WBS. Cumulative mode partitioning.

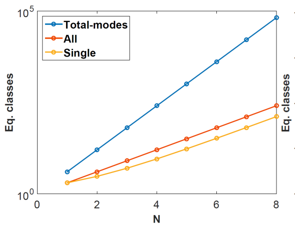


**Fig. 13.** SRCN. Largest mode partition.

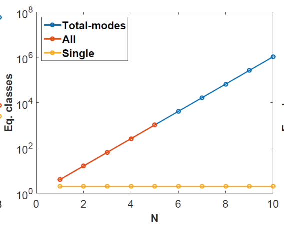


**Fig. 14.** NLTL. Largest mode partition.

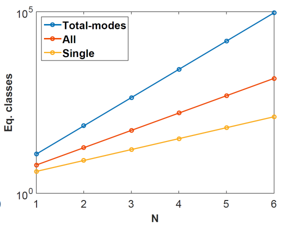


**Fig. 15.** WBS. Largest mode partition.

| | N | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| SRCN | Modes | 4 | 16 | 64 | 256 | 1024 | 4096 | 16384 | 65536 | | |
| SRCN | Input Vars U | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | |
| SRCN | State Vars X | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | | |
| SRCN | Ref. Vars Y | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | | |
| NLTL | Modes | 4 | 16 | 64 | 256 | 1024 | 4096 | 16384 | 65536 | 262144 | 1048576 |
| NLTL | Input Vars U | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| NLTL | State Vars X | 4 | 8 | 12 | 16 | 20 | 24 | 28 | 32 | 36 | 40 |
| NLTL | Ref. Vars Y | 4 | 8 | 12 | 16 | 20 | 24 | 28 | 32 | 36 | 40 |
| WBS | Modes | 8 | 32 | 128 | 512 | 2048 | 8192 | | | | |
| WBS | Input Vars U | 1 | 1 | 1 | 1 | 1 | 1 | | | | |
| WBS | State Vars X | 2 | 3 | 4 | 5 | 6 | 7 | | | | |
| WBS | Ref. Vars Y | 3 | 4 | 5 | 6 | 7 | 8 | | | | |

**Table 4.** Benchmark sizes.