SMT-based Analysis of Switching Multi-Domain Linear Kirchhoff Networks

Alessandro Cimatti
Fondazione Bruno Kessler
Trento, IT
Email: cimatti@fbk.eu

Sergio Mover
University of Colorado Boulder
Boulder, USA
Email: sergio.mover@colorado.edu

Mirko Sessa
Fondazione Bruno Kessler and
University of Trento
Trento, IT
Email: sessa@fbk.eu

***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 Linear Kirchhoff Networks (SMDLKN). In this paper, we tackle a major obstacle to formal verification of SMDLKN, 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.

I. Introduction

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 [1], depicted in Fig. 1, where the pressure applied by a hydraulic circuit (including valves and pumps) operates moving components from the hydro-mechanical domain (e.g. a cylinder). Basic components (e.g. valves, accumulators, and tanks) 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 Linear Kirchhoff Networks (SMDLKN) [2]. 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.

In this paper, we investigate methods for the formal analysis of SMDLKN, 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

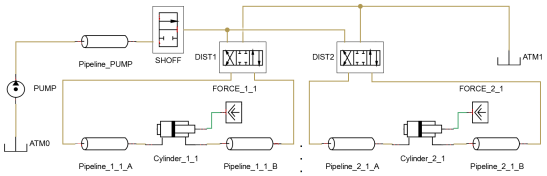

Fig. 1. Landing Gear System with $N = 2$ hydraulic cylinder lines ($\text{LGS}_{[N]}$).

that is amenable to formal verification. The existing formal verification tools for hybrid systems [3], [4], [5] 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* [6]. 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, 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.

In this paper, we propose a general method to reformulate SMDLKN 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* [7] 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.

We adopt an approach based on Satisfiability Modulo Theories (SMT) [8] 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.

We perform an experimental evaluation on several multi-domain scalable real-world benchmark applications. The proposed optimizations substantially increase the scalability of the procedures, allowing us to validate and reformulate SMDLKN 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 [5]).

The rest of the paper is organized as follows. Section II provides the necessary background notions. Section III defines the validation and reformulation problems for SMDLKN, Section IV presents the proposed symbolic algorithms. Section V discusses the related work. Section VI presents the experimental evaluation. We conclude in Section VII.

II. BACKGROUND

A. Notation

$|X|$ denotes the cardinality of the set X. Given a set of real variables X, the notation $\vec{X}$ refers to the vector that contains all the variables in X ordered lexicographically. $\mathbb{R}$, $\mathbb{R}_{\geq 0}$, $\mathbb{B}$ denote the set of Real numbers, non-negative Real numbers, and Boolean. If X is a set of variables, then X' and $\dot{X}$ are the sets obtained by replacing each element with its primed and dotted version, resp.

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) [8]. Given a first-order logic formula ψ and a set of variables X, $\psi(X)$ denotes that X is the set of free variables in ψ. $\varphi \models_{\mathcal{T}} \psi$ denotes that the formula ψ is a logical consequence of φ in the theory $\mathcal{T}$; when clear from context, we omit $\mathcal{T}$ and simply write $\varphi \models \psi$. An assignment μ for a set of variables X is the set $\{x \mapsto c \mid x \in X \text{ and } c \text{ is a constant}\}$, $\mu_{|X}$ is the projection of all the assignments in μ only onto the variables contained in X, and $\mu(x)$ is the value assigned to x in μ. Abusing the notation, we interchange the linear system notation (e.g. $\vec{X} = \vec{B}\vec{Y}$, where $|\vec{X}| = n \times 1, |\vec{B}| = n \times m, |\vec{Y}| = m \times 1$) with the conjunction of predicates in LRA corresponding to the matrix product (e.g. $\bigwedge_{i=1}^{n} \vec{X}[i] = \sum_{j=1}^{m} (\vec{B}[i][j]\vec{Y}[j])$). Given two vectors (resp. matrices) A and B, $A \cdot B$ denotes their vertical (resp. horizontal) concatenation.

B. Linear Systems

The linear system $\vec{A}\vec{W} = \vec{b}$ is *homogeneous* if $\vec{b} = \vec{0}$, and, in that case, it admits at least the solution $\vec{W} = \vec{0}$.

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

Lemma 1 (Linearity [9]): 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. ■

C. Hybrid Automata

Hybrid automata (HA) [10] represent a system with continuous and discrete dynamics. We use a symbolic representation of hybrid automata, where the discrete locations and transitions are represented by means of SMT formulae [11].

A HA is a tuple $H = \langle V, X, Init, Invar, Trans, Flow \rangle$ where 1) V is the set of discrete variables; 2) X is the set of continuous variables; 3) $Init(V, X)$ represents the set of initial states; 4) $Invar(V, X)$ represents the set of invariant states; 5) $Trans(V, X, V', X')$ represents the set of discrete transitions; 6) $Flow(V, \dot{X}, X)$ is the flow condition. We assume that all the formulas $Init$, $Invar$, $Trans$ and $Flow$ are quantifier-free and linear.

In the above definition, $Flow$ may either define a system of Differential-Algebraic Equations (DAEs) or Ordinary Differential Equations (ODEs). We say that the automaton has an *ODE dynamics* if, for each assignment μ to V, $Flow$ is equivalent to a system of ODEs (i.e. $\dot{\vec{X}} = \vec{A}\vec{X}$). Otherwise, the automaton has a *DAE dynamics*.

A *state* of a hybrid automaton is an assignment to the variables $V \cup X$, and a *run* is a sequence of states such that the first state is in the initial states, every state belongs to the invariant, and each pair of consecutive states either satisfies $Trans$ or the solution to the differential equations described in the $Flow$ condition. The semantics of the HA is defined by the runs that it accepts. Two hybrid automata H_1 and H_2 are *equivalent* if they accept the same runs.

D. Switching Multi-Domain Linear Kirchhoff Networks

Definition 1 (Network component): A component c_i is a tuple $\langle B_i, R_i, T_i, invar_i,\ flow_i, input_i, trans_i \rangle$ where:

- B_i : set of *discrete variables* representing the modes.
- R_i : set of *continuous variables* representing the physical quantities of the component. We partition the set of continuous variables in three disjoint sets of *state* (X_i), *input* (U_i) and *output* (Y_i) variables.
- $invar_i : 2^{B_i} \rightarrow 2^{P_{red}}$: *invariant* conditions, where P_{red} is a set of predicates over the variables R_i.
- $input_i : 2^{U_i} \rightarrow 2^{\mathcal{F}_i}$: *input binding* assigning a continuous function of time ($\mathcal{F}_i = \{f(t) | f \text{ is continuous}\}$) to each input variable in U_i.
- $flow_i : 2^{B_i} \rightarrow 2^{P_{eq}}$: *flow* condition, where P_{eq} is a set of homogeneous linear equalities with variables from $X_i, U_i, Y_i, \dot{X}_i$.
- $trans_i(B_i, R_i, B_i')$: *mode transition* condition that represents the mode transitions (with guards) that can happen in the component.

Definition 2: A *Switched Multi-Domain Linear Kirchhoff Network* (SMDLKN) $\mathcal{N}$ is a tuple $\langle \mathcal{C}, K \rangle$, where $\mathcal{C}$ is a set of components and K is a set of equalities among continuous variables of the components, that represents the Kirchhoff conservation rules (i.e. the set of connection constraints).

We extend the notation used to specify the set of component variables to a network $\mathcal{N}$, defining the sets $B := \bigcup_{c_i \in \mathcal{C}} B_i, R := \bigcup_{c_i \in \mathcal{C}} R_i, \ldots$. Let $V = B \cup R$ be the set of all the variables of a network. A *state* of the network is given by an assignment μ to *all the variables* V. We refer to each possible (complete) assignment $\mu_b \in 2^B$ to *all the discrete variables* B as a *mode* of the network. Every different network mode induces a continuous dynamics described by a DAE.

Definition 3 (Differential-Algebraic Equation of a mode): The DAE $DAE(\mu_b)$ of a mode μ_b is defined as the set of constraints:

$$DAE(\mu_b) := \bigcup_{c_i \in \mathcal{C}} flow_i(\mu_{b|B_i}) \cup K \quad (1)$$

$DAE(\mu_b)$ can be equivalently represented as a linear system:

$$\vec{M}\dot{\vec{X}} + \vec{N}\vec{X} + \vec{O}\vec{Y} + \vec{P}\vec{U} = \vec{0} \quad (2)$$

for some coefficient matrices $\vec{M} \in \mathbb{R}^{l \times |X|}, \vec{N} \in \mathbb{R}^{l \times |X|}$, $\vec{O} \in \mathbb{R}^{l \times |Y|}, \vec{P} \in \mathbb{R}^{l \times |U|}$, and a positive integer l equal to the number of system constraints.

Definition 4 (Network semantics): The semantics of the network $\mathcal{N}$ is the hybrid automaton $H_{\mathcal{N}} = \langle V_H, X_H, Init, Invar, Trans, Flow \rangle$ where

$$V_H := B \quad X_H := X \cup U \cup Y \quad Init(V_H, X_H) := True$$

$$Invar(V_H, X_H) := \bigwedge_{\mu_b \in 2^B} (\mu_b \rightarrow \bigwedge_{c_i \in \mathcal{C}} invar_i(\mu_{b|B_i}))$$

$$Trans(V_H, X_H, V'_H, X'_H) := \bigwedge_{c_i \in \mathcal{C}} trans_i \wedge \bigwedge_{x \in X} (x' = x)$$

$$Flow(V_H, \dot{X}_H, X_H) := \bigwedge_{\mu_b \in 2^B} (\mu_b \rightarrow DAE(\mu_b)) \wedge \bigwedge_{c_i \in \mathcal{C}} input_i(U_i)$$

III. Validation and Reformulation problems

Given a network $\mathcal{N} = \langle \mathcal{C}, K \rangle$, our first goal is to automatically check if it contains inconsistencies, which represent an unwanted condition in the real system modeled by the network.

Definition 5: A mode μ_b of a network $\mathcal{N}$ is *consistent* if, for every possible assignment to the state (X) and input (U) variables, the linear system $DAE(\mu_b)$ has at least a solution.

An inconsistent mode in the network represents an undesired condition in the physical system that must be avoided. Consider the electrical circuit of Fig. 2, where the voltages V_{C_1} and V_{C_2} across the capacitors C_1 and C_2 are state variables, and the current I_B imposed by the current generator B is the input variable (we use I and V to refer to currents and voltages, and we use the component's name as subscript to identify the current or voltage of that component). For the sake of brevity, all the electrical parameters take value one and have been omitted from the following formulas. The DAE associated to the discrete mode where both the switches S_1 and S_2 are open is $I_B = I_R, I_R = I_{S_1} + I_{S_2}, I_{S_1} = 0, I_{S_2} = 0, I_{C_1} = \dot{V}_{C_1}, I_{C_2} = \dot{V}_{C_2}$. The mode is not consistent when $I_B \neq 0$. 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 $\mathcal{N}$. As explained in the introduction, a requirement imposed by the symbolic verification tools for hybrid systems (e.g. HYCOMP [5]) is 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 $DAE(\mu_b)$ as a system of *Ordinary Differential Equations*: $\dot{\vec{X}} = \vec{A}\vec{X} + \vec{B}\vec{U}$, 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 values of the state and input variables, returns one and unique value for the first derivative variables $\dot{X}$. Consistency is a necessary condition to ensure the existence of the ODE function, while the other necessary condition is the determinicity of the values assigned to $\dot{X}$.

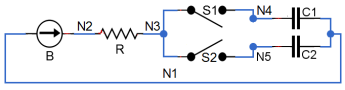

Fig. 2. Schematic of a simple electrical circuit.

Definition 6: A mode μ_b of a network $\mathcal{N}$ is *deterministic* if, for every possible value of the state X and input variables U, the linear system $DAE(\mu_b)$ admits at most one solution of the first derivative $\dot{X}$.

In the example of Fig. 2, the DAE of the discrete mode where both the switches S_1 and S_2 are closed is $I_B = I_R, I_R = I_{S_1} + I_{S_2}, I_{C_1} = I_{S_1}, I_{C_2} = I_{S_2}, I_{C_1} + I_{C_2} = I_B, I_{C_1} = \dot{V}_{C_1}, I_{C_2} = \dot{V}_{C_2}$. In this DAE, the values of the currents I_{C_1} and I_{C_2} are not uniquely identified when fixing a value of the input I_B (e.g. if $I_B = 3$ then the only constraints for I_{C_1} and I_{C_2} is that $I_{C_1} + I_{C_2} = 3$), and hence also the values of $\dot{V}_{C_1}$ and $\dot{V}_{C_2}$ is not. Due to this non-determinism, the above DAE cannot be rewritten as an ODE.

Definition 7: A mode μ_b of a network $\mathcal{N}$ is *valid* if it is both *consistent* and *deterministic*. The network $\mathcal{N}$ is *valid* if all the modes $\mu_b \in 2^B$ are valid.

Definition 8 (Validation problem): Given a network $\mathcal{N}$, the validation problem consists of deciding if $\mathcal{N}$ is valid.

Remark 1: We note that a mode is valid if it is associated to a *index-1* DAE, while it is invalid for *higher-index* DAEs.

Definition 9 (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.

In the electrical circuit in Fig. 2, the mode where the switch S_1 is closed and S_2 is open has the DAE $I_B = I_R, I_R = I_{S_1}, I_{S_1} = I_{C_1}, I_{C_1} = \dot{V}_{C_1}, 0 = I_{S_1}, I_{S_2} = I_{C_2}, I_{C_2} = \dot{V}_{C_2}$. The mode is valid, and the ODE representation of the DAE is $\dot{V}_{C_1} = 0V_{C_1} + 0V_{C_2} + 1I_B, \dot{V}_{C_2} = 0V_{C_1} + 0V_{C_2} + 0I_B$.

IV. Symbolic validation and reformulation

A. 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 $\mathcal{N} = \langle \mathcal{C}, K \rangle$:

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 $\mu_b \in 2^B$ 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.

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

$$\psi_{\text{DAE}} := (\bigwedge_{c_i \in \mathcal{C}} \bigwedge_{\mu_b \in 2^B i} (\mu_b \rightarrow flow_i(\mu_b))) \wedge \bigwedge_{k \in K} k \quad (3)$$

ψ_{DAE} predicates over the same variables of the network, so we reuse the same notation introduced in Sec. II-D 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 components local modes in place of the network global modes, thus preventing the blow-up of the formula ψ_{DAE}.

Lemma 2: μ is a satisfying assignment of ψ_{DAE} iff $\mu_{|R}$ is a solution of $DAE(\mu_{|B})$ ■

We provide the proofs of the lemmas and theorems in an extended version of the paper available at http://es.fbk.eu/people/sessa/paper/fmcad17/main.pdf

b) 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.

c) 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.

d) Reformulating the network: We reformulate a valid network $\mathcal{N}$ into the hybrid automaton $H^r_{\mathcal{N}} = \langle V^r, X^r, Init^r, Invar^r, Trans^r, Flow^r \rangle$. $H^r_{\mathcal{N}}$ 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 $Invar^r := \psi_Y \wedge \bigwedge_{c_i \in \mathcal{C}} \bigwedge_{\mu_b \in 2^B i} (\mu_b \rightarrow invar_i(\mu_b))$, while $Flow^r := \psi_{\dot{X}} \wedge \bigwedge_{c_i \in \mathcal{C}} input_i(U_i)$. The formula $\psi_{\dot{X}}$ is the reformulation of the variables $\dot{X}$, while ψ_Y is a relation that represents the values of the output 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).

B. Optimized Validation and Reformulation

We improve the basic validation and reformulation procedures by applying an extension of the *implicit function theorem* [9]. 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 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 output 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 output variables Y that, for example, can assume a value non-deterministically. For this reason we extend the implicit function theorem as follows.

a) Implicit Function Theorem:

Theorem 1 (Implicit Function Theorem): Let m, n, l be positive integers. Let $F : \mathbb{R}^{m+n} \rightarrow \mathbb{R}^l$ 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 $\vec{W}_h$ such that its j-th component w_j is different from zero

iff there exists a unique linear function $f_j : \mathbb{R}^n \rightarrow \mathbb{R}^1$ 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 $DAE(\mu_b)$ of the mode μ_b and its matrix representation $\vec{M}\dot{\vec{X}} + \vec{N}\vec{X} + \vec{O}\dot{\vec{Y}} + \vec{P}\vec{U} = \vec{0}$ (see Equation 2). One can directly apply Theorem 1, just by noticing that $DAE(\mu_b)$ is indeed a linear homogeneous implicit function $F(\vec{W}, \vec{Z})$, where $\vec{W} := \dot{\vec{X}} \cdot \vec{Y}$, $\vec{Z} := \vec{U} \cdot \vec{X}$, $\vec{A} := \vec{M} \cdot \vec{O}$, and $\vec{B} := \vec{P} \cdot \vec{N}$. If the first condition of Theorem 1 holds for all the columns $\vec{b_i}$ of the concatenated coefficient matrix $\vec{B} := \vec{P} \cdot \vec{N}$, then μ_b is consistent, while if the second condition holds for all $\dot{x} \in \dot{X}$, then μ_b is deterministic. Then, if both conditions hold, the mode μ_b is valid.

b) 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 IV-A. We achieve this by directly checking the conditions of Theorem 1. The consistency condition (1) of the Theorem 1 is encoded as:

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

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

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

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 $\vec{B} := \vec{P} \cdot \vec{N}$ in order to symbolically check the conditions of the Theorem 1.

Lemma 3: A network $\mathcal{N}$ is consistent iff for all $\mu_b \in 2^B$, $\mu_b \models \psi_{con}(B)$, and is deterministic iff for all the modes $\mu_b \in 2^B$, $\mu_b \models \psi_{det}(B)$ ■

c) Reformulation: The algorithm PERVARIABLEREF (Fig. 3) synthesizes the formulas $\psi_{\dot{X}}$ and ψ_Y used in the reformulated automaton $H_{\mathcal{N}}^r$, by using Theorem 1 and Lemma 1.

In the algorithm, we use the SMT solver primitives *push*, *assert*, *isSat*, *pop*, *reset* (see e.g. [12]), *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 (Fig. 5) 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 $\dot{X}_s = \{\dot{x}_s \mid \dot{x} \in \dot{X}\}$). $\psi_{Y,\dot{x}}$ 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 1 to synthesize the reformulation. According to Lemma 1, we know that, for a mode $\mu_b \in 2^B$ and a variable $\dot{x}$, the function $f_{\dot{x}}(\vec{U} \cdot \vec{X})$ such that $\dot{x} = f_{\dot{x}}(\vec{U} \cdot \vec{X})$ is defined as $f_{\dot{x}}(\vec{U} \cdot \vec{X}) := \bar{\vec{W}}_{p_1}^j z_1 + ... + \bar{\vec{W}}_{p_n}^j z_n$, where j is the index corresponding to the variable $\dot{x}$ in the vector $\vec{W} := \dot{\vec{X}} \cdot \vec{Y}$, $\bar{\vec{W}}_{p_i}^j$ 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 $f_{\dot{x}}(\vec{U} \cdot \vec{X})$ by computing all the n *particular* solutions of the system and taking their j-th element. Fig. 5 shows the reformulation procedure for a single variable $\dot{x}$: each execution of the loop at Line 4 finds a mode $\mu_b \in 2^B$ (Line 5) for which the $\dot{x}$ reformulation is still unknown. Then (Line 6) the algorithm computes the coefficients D of the $\dot{x}$ reformulation in μ_b. 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 $\vec{U} \cdot \vec{X}$. 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 $\mu_b \in 2^B$.

GETCOEFF is shown in Fig. 6. For each variable $z_i \in U \cup X$, the condition built at Line 4 reduces the term $\vec{B}(\vec{U} \cdot \vec{X})$ 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 (Fig. 4) builds the condition γ that is satisfiable in every $\mu_b \in 2^B$ 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 $H_{\mathcal{N}}^r$ is equivalent to the hybrid automaton $H_{\mathcal{N}}$ that defines the network semantics. ■

V. RELATED WORK

Multi-Domain Linear Kirchhoff Networks are widely used in various engineering applications [13], [14], [15]. Different tools support the acausal modeling phase [16], [17], 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 [6], [18], [19], 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. [17]) restrict the use of components

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

Fig. 3. Reformulation algorithm for $\mathcal{N}$.

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

Fig. 4. Find the cluster of modes that share the same coefficients D for $\dot{x}$.

REFORM (ψ_{DAE}, X, U, $\dot{x}$):
1. $Ref_{\dot{x}} := True$
2. $\psi_{Y,\dot{x}} := True$
3. solver.assert($True$)
4. **while** solver.isSat():
 # Get a fresh mode
5. $\mu_b :=$ solver.getModel()
 # Get the row vector of coeff. that
 # contributes to $\dot{x}$ in μ_b
6. $D :=$ GETCOEFF (ψ_{DAE}, X, U, $\dot{x}$, μ_b)
 # Get the cluster of modes that share the
 # same coeff.
7. $\beta :=$ GETEQMOD (ψ_{DAE}, X, U, $\dot{x}$, D)
 # Prune the cluster of modes from the search
8. solver.assert($\neg\beta$)
 # Build the reformulation equation
9. $Eq := \dot{x} = D\ (\ \vec{U} \cdot \vec{X})$
10. $Ref_{\dot{x}} := Ref_{\dot{x}} \wedge (\beta \rightarrow Eq)$
11. $\psi_{Y,\dot{x}} := \psi_{Y,\dot{x}} \wedge \beta \rightarrow \dot{x}_s = D(\vec{U} \cdot \vec{X})$
12. **return** $(Ref_{\dot{x}}, \psi_{Y,\dot{x}})$

Fig. 5. Reformulation of a single variable $\dot{x}$.

GETCOEFF (ψ_{DAE}, X, U, $\dot{x}$, μ_b):
 # D row vector of coeff. w.r.t. $U \cup X$
1. coeffSolver.assert($\psi_{\text{DAE}} \wedge \mu_b$)
2. **for each** $z_i \in U \cup X$:
3. coeffSolver.push()
 # build the rhs corresponding to z_i
4. $rhs_{z_i} := z_i = 1 \wedge \bigwedge_{l \in (U \cup X) \setminus \{z_i\}} l = 0$
5. coeffSolver.assert(rhs_{z_i})
 # get a system solution
6. $\mu' :=$ coeffSolver.getModel()
 # $\mu'(\dot{x})$ is the coeff. w.r.t. z_i
7. $D[i] := \mu'(\dot{x})$
8. coeffSolver.pop()
9. **return** D

Fig. 6. Computes the ref. coefficients D of $\dot{x}$.

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

The closest related work is [20], that presents a method to convert Switched *Electrical* Linear Kirchhoff Networks (SELKN) into hybrid automata. The work proposed here is more general than [20] in three respects. First, we are able to deal with *multi-domain* networks, enabling mechanical, electrical and hydraulic domains, and their combination, whilst [20] is restricted to electrical networks. Second, the method in [20] 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 SMDLKN with *non-deterministic* output 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 [20]. We remark that all the experiments presented in the next section are based on benchmarks that are out of reach for the method in [20]. In [21], 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 [21] (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 [21], 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 [22], 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 SMDLKN, 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 [23] do not directly consider Multi-Domain Linear Kirchhoff Network, but work on hybrid automata [10]. Tools like *SpaceEx* [3] or *Flow** [24] 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 [25], [5], [26], [27] reason on the symbolic representation of the system. HYBRIDSAL [25] and HYCOMP[5] analyze linear hybrid systems whose continuous dynamics is specified with a linear ODE. DREACH [26], [27] 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 [28], [29], [30], [31], [32]. 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 [33], [34]. This causal semantics considers systems represented as a connection of input-output functional blocks, posing a major obstacle to the modeling of SMDLKN. 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.

VI. EXPERIMENTAL EVALUATION

Setup: We implemented the proposed approach using the PYSMT [35] library and the MATHSAT5 [12] SMT-solver. At the core, we use the symbolic model checker HYCOMP [5]. The resulting workflow takes as input a SMDLKN 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 IV-A, while OPT refers to

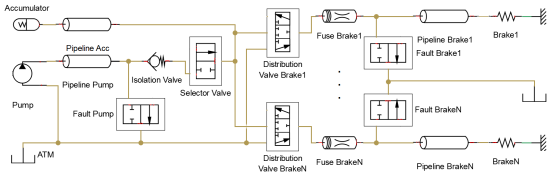

Fig. 7. Wheel Braking System, Arch.2, with N braking lines ($\text{WBSA2}_{[N]}$)

those of Section IV-B. We run the experiments on a 3.5 GHz cpu with 16GB RAM, with time out (TO) set to 3600s for VAL, 43200s for REF, and 18000s for VER. The tools and the benchmarks are available at http://es.fbk.eu/people/sessa/attachment/fmcad17/fmcad17.tar.bz2.

Benchmarks: We consider five scalable benchmarks: two variants of the Wheel Braking System (WBS) from the SAE standard AIR6110 [36], the Landing Gear System (LGS), and two variants of a hydraulic tank network (WW) inspired from [21], for a total of 29 instances. The WBS benchmarks, $\text{WBSA2}_{[N]}$ and $\text{WBSA4}_{[N]}$, are parameterized w.r.t. the number N of braking lines (see Fig. 7). The benchmarks differ in the position of the hydraulic accumulator line. Fig. 1 shows (part of) the Landing Gear System ($\text{LGS}_{[N]}$) from [1], which is parameterized w.r.t. the number N of cylinder lines. The $\text{WWLIN}_{[N]}$ and $\text{WWRING}_{[N]}$ benchmarks represent networks of N hydraulic tanks connected either linearly or in a ring through channels, composed by pipes and valves. The WW benchmarks are originally proposed in [21], with a hand-crafted technique meant for the automatic generation of hybrid benchmarks that abstracts away the mutual interactions among the liquid levels stored in the tanks. On the contrary, our work aims at faithfully representing the physics of the real system. Our SMDLKN-based models capture the physical dynamics of the (bidirectional) flow through the channels, and naturally represents the global interaction of the interconnected components, retaining the compositional structure of the physical system.

The features of the benchmarks are described in the extended version of the paper. The models contain tens of boolean variables and hundreds of real variables, resulting in up to 2 millions of modes. None of the benchmarks considered in this evaluation can be analyzed with the approach presented in [20]. There are several reasons for this. First, all the benchmarks are out of the electrical domain. Even if [20] deals with some simple hydraulic models by means of the hydraulic-electrical analogy, the cylinder component used in the LGS does not fit in the domain analogy. Second, [20] cannot deal with non-deterministic output variables. Our WBS benchmarks yield under-specified output variables that were not present in the much simpler and less complete model used in [20]. Finally, the WW benchmarks contain some inconsistent modes, and the method in [20] requires consistency for all the modes.

Also, note that our modeling of the WBS benchmarks is different than the model presented in [37], which is an abstract, discretized and causal model of the system suitable to perform a formal system safety assessment analysis. Instead, in our WBS model we capture the real continuous physics of the system.

Validation: The results of the evaluation are summarized in Tab. I. First, we consider the runtime of the basic (BAS) and the optimized (OPT) encodings for validation. We see that OPT solves all the 29 instances, while BAS times out on the 10 biggest instances. Focusing on the instances solved by both encodings, OPT outperforms BAS by two orders of magnitude and scales much better w.r.t the benchmark size. Noteworthy, the OPT method validates the two millions of modes of the $\text{WBS}_{[5]}$ instances within 327 and 252 seconds, respectively. 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 problem.

All the WBS and LGS benchmarks have only consistent modes. This does not hold for the WW benchmarks, where a tank cannot accept incoming [respectively, provide outgoing] liquid in mode full [resp., empty]. Notice that the full and empty modes can be seen as hazardous configurations of the network, when an actuator must pump in/out a fluid. Our validation approach is able to detect and report such bad configurations, and allows us to generate models under the assumption that the invalid modes are not entered (e.g. by the preventive action of a supervisory controller).

	VAL		REF		VER
	BAS	OPT	BAS	OPT	OPT
$\text{LGS}_{[2]}$	1	0	187	1	1
$\text{LGS}_{[3]}$	5	1	TO	5	1
$\text{LGS}_{[4]}$	29	3	TO	21	1
$\text{LGS}_{[5]}$	204	9	TO	90	7
$\text{LGS}_{[6]}$	1567	25	TO	449	9
$\text{LGS}_{[7]}$	TO	73	TO	3091	14
$\text{LGS}_{[8]}$	TO	215	TO	30269	30
$\text{WBSA2}_{[2]}$	10	0	TO	3	0
$\text{WBSA2}_{[3]}$	395	5	TO	19	4
$\text{WBSA2}_{[4]}$	TO	37	TO	204	74
$\text{WBSA2}_{[5]}$	TO	327	TO	5554	2630
$\text{WBSA4}_{[2]}$	9	0	TO	3	0
$\text{WBSA4}_{[3]}$	360	4	TO	22	5
$\text{WBSA4}_{[4]}$	TO	30	TO	223	131
$\text{WBSA4}_{[5]}$	TO	252	TO	5892	10970
$\text{WWLIN}_{[2]}$	0	0	8	0	0
$\text{WWLIN}_{[3]}$	0	0	1072	1	0
$\text{WWLIN}_{[4]}$	2	0	TO	3	2
$\text{WWLIN}_{[5]}$	21	0	TO	8	5
$\text{WWLIN}_{[6]}$	166	1	TO	19	33
$\text{WWLIN}_{[7]}$	1670	3	TO	53	62
$\text{WWLIN}_{[8]}$	TO	5	TO	419	343
$\text{WWRING}_{[2]}$	0	0	39	0	0
$\text{WWRING}_{[3]}$	4	0	TO	3	1
$\text{WWRING}_{[4]}$	74	1	TO	10	6
$\text{WWRING}_{[5]}$	1300	3	TO	30	28
$\text{WWRING}_{[6]}$	TO	7	TO	89	78
$\text{WWRING}_{[7]}$	TO	15	TO	369	848
$\text{WWRING}_{[8]}$	TO	27	TO	2465	MO

TABLE I
VALIDATION, REFORMULATION AND VERIFICATION TIME [S].

Reformulation: We consider the runtime for the BAS reformulation lower bound, and the OPT reformulation. The BAS encodings cannot deal with the benchmarks, whereas the OPT encodings successes in reformulating all the instances. Again, this happens because the OPT encodings exploits the properties of the algebraic structure of the problem to mitigate the computational complexity of the quantifier elimination in the computation of the derivative variables reformulation. Additionally, the variable 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 output variables.

We notice that the reformulation of the WW benchmarks is restricted to the *valid* modes, while considering the *non-valid* modes as a macro error state of the network. The ability of representing these non-valid modes in the reformulated hybrid automaton is crucial when considering the functional verification of the network composed with a controller designed to prevent the reachability of hazardous configurations.

Verification: For both WBS benchmarks we consider the property P_1: *when the selector valve is closed, a brake command cannot actuate any brake*. Consistently with the SAE standard AIR6110 [36], that describes such design flaw, P_1 is violated for $\text{WBSA2}_{[N]}$ and is verified by $\text{WBSA4}_{[N]}$. For the LGS, we consider the (false) property L_1: *the first cylinder cannot reach its end-of-stroke*. For both WW benchmarks, we consider the (false) property W_1: *the level of the first tank cannot exceed a given threshold*, that is violated closing all the valves connected to the first tank.

The verification on the hybrid automata from the OPT reformulation completes within the time out on all the benchmarks, returning the expected results, except for $\text{WWRING}_{[8]}$ that experienced a memory out (MO). 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 SMDLKN.

VII. CONCLUSION

We presented an SMT-based method for the formal analysis of Switching Multi-Domain Linear Kirchhoff Networks (SMDLKN), that is able to automatically validate and reformulate a SMDLKN 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.

We plan to extend the approach to incorporate networks with discontinuous state variables [38], produce a network of HA instead of a monolithic HA, and extend the analysis towards the safety assessment for the generation of Fault Trees.

REFERENCES

[1] F. Boniol and V. Wiels, *The Landing Gear System Case Study*. Cham: Springer International Publishing, 2014.

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

[3] G. Frehse, C. L. 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*, 2011, pp. 379–395.

[4] S. Gao, S. Kong, and E. M. Clarke, "dreal: An SMT solver for nonlinear theories over the reals," in *CADE-24*, 2013.

[5] A. Cimatti, A. Griggio, S. Mover, and S. Tonetta, "HyCOMP: An SMT-based model checker for hybrid systems," in *TACAS*, 2015.

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

[7] J. R. Munkres, *Analysis on manifolds*. Westview Press, 1997.

[8] C. W. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli, "Satisfiability modulo theories," in *Handbook of Satisfiability*, 2009.

[9] S. J. Axler, *Linear Algebra Done Right*, ser. Undergraduate Texts in Mathematics. New York: Springer, 1997.

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

[11] A. Cimatti, S. Mover, and S. Tonetta, "A quantifier-free SMT encoding of non-linear hybrid automata," in *FMCAD*, 2012, pp. 187–195.

[12] A. Cimatti, A. Griggio, B. J. Schaafsma, and R. Sebastiani, "The mathsat5 smt solver," in *TACAS*. Springer Berlin Heidelberg, 2013.

[13] H. Yoshikawa, T. Oda, K. Nonaka, and K. Sekiguchi, "Modeling and simulation for leg-wheel mobile robots using modelica," in *The First Japanese Modelica Conferences*, no. 124, 2016.

[14] 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.

[15] A. M. Krishna, Arash Dizqah and B. P. Fritzson, "Modeling and simulation of a combined solar and wind systems using openmodelica," 2013.

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

[17] T. Mathworks, "Simscape power systems." [Online]. Available: http://it.mathworks.com/help/physmod/sps/index.html

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

[19] D. L. Skaar, "Using the superposition method to formulate the state variable matrix for linear networks," *IEEE Transactions on Education*, vol. 44, no. 4, Nov 2001.

[20] A. Cimatti, S. Mover, and M. Sessa, "From electrical switched networks to hybrid automata," in *FM 2016*, vol. 9995, 2016.

[21] S. Bak, S. Bogomolov, M. Greitschus, and T. T. Johnson, "Benchmark generator for stratified controllers of tank networks," in *ARCH*, 2015.

[22] N. Fulton, S. Mitsch, J. Quesel, M. Völp, and A. Platzer, "Keymaera X: an axiomatic tactical theorem prover for hybrid systems," in *CADE*, 2015.

[23] R. Alur, "Formal verification of hybrid systems," in *EMSOFT 2011*, 2011.

[24] X. Chen, E. Ábrahám, and S. Sankaranarayanan, "Flow*: An analyzer for non-linear hybrid systems," in *CAV*. Springer, 2013.

[25] A. Tiwari, "HybridSAL Relational Abstracter," in *CAV*, 2012.

[26] S. Kong, S. Gao, W. Chen, and E. M. Clarke, "dReach: δ-reachability analysis for hybrid systems," in *TACAS*, 2015.

[27] K. Bae, S. Kong, and S. Gao, "SMT encoding of hybrid systems in dReal," in *ARCH14-15*. EasyChair, 2015.

[28] M. H. Zaki, S. Tahar, and G. Bois, "Formal verification of analog and mixed signal designs: Survey and comparison," in *2006 IEEE North-East Workshop on Circuits and Systems*, June 2006, pp. 281–284.

[29] T. Dang, A. Donzé, and O. Maler, "Verification of analog and mixed-signal circuits using hybrid system techniques," in *FMCAD 2004*, 2004.

[30] G. Frehse, B. H. Krogh, R. A. Rutenbar, and O. Maler, "Time domain verification of oscillator circuit properties," *Electr. Notes Theor. Comput. Sci.*, vol. 153, no. 3, pp. 9–22, 2006.

[31] H. L. Lee, M. Althoff, S. Hoelldampf, M. Olbrich, and E. Barke, "Automated generation of hybrid system models for reachability analysis of nonlinear analog circuits," in *ASP-DAC 2015*, 2015.

[32] Y. Zhang, S. Sankaranarayanan, and F. Somenzi, "Piecewise linear modeling of nonlinear devices for formal verification of analog circuits," in *FMCAD*, 2012, pp. 196–203.

[33] K. Manamcheri, S. Mitra, S. Bak, and M. Caccamo, "A step towards verification and synthesis from simulink/stateflow models," in *HSCC 2011*, 2011, pp. 317–318.

[34] S. Minopoli and G. Frehse, "SL2SX Translator: From Simulink to SpaceEx Models," in *HSCC 2016*, 2016, pp. 93–98.

[35] M. Gario and A. Micheli, "pySMT: a Solver-Agnostic Library for Fast Prototyping of SMT-Based Algorithms," in *SMT Workshop*, 2015.

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

[37] M. Bozzano and et al., "Formal design and safety analysis of AIR6110 wheel brake system," in *CAV*, 2015.

[38] 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.