

From Hybrid Automata to DAE-Based Modeling

Albert Benveniste, Benoît Caillaud, Mathias Malandain

▶ **To cite this version:**

Albert Benveniste, Benoît Caillaud, Mathias Malandain. From Hybrid Automata to DAE-Based Modeling. Principles of Systems Design, 13660, Springer Nature Switzerland, pp.3-20, 2022, Lecture Notes in Computer Science, 10.1007/978-3-031-22337-2_1 . hal-03921708

HAL Id: hal-03921708
https://inria.hal.science/hal-03921708

Submitted on 4 Jan 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution| 4.0 International License

From Hybrid Automata to DAE-based modeling

Albert Benveniste[1], Benoît Caillaud[1], and Mathias Malandain[1]

Inria centre at Rennes University
Campus de Beaulieu, 35042 Rennes Cedex, France
email: firstname.lastname@inria.fr

Abstract. Tom Henzinger was among the co-founders of the paradigm of *hybrid automata* in 1992. Hybrid automata possess different locations, holding different ODE-based dynamics; exit conditions from a location trigger transitions, resulting in starting conditions for the next location. A large research activity was developed in the formal verification of hybrid automata; this paradigm still grounds popular commercial tools such as Stateflow for Simulink.
However, modeling from first principles of physics requires a different approach: balance equations and conservation laws play a central role, and elementary physical components come with no prespecified input/output profile. All of this leads to grounding physical modeling on DAEs (Differential Algebraic Equations, of the form $f(x', x, v) = 0$) instead of ODEs. DAE-based modeling, implemented for example in the Modelica language, allows for modularity and reuse of models.
Unsurprisingly, DAE-based hybrid systems (also known as *multimode DAE systems*) emerge as the central paradigm in multiphysics modeling. Despite the growing popularity of modeling tools based on this paradigm, fundamental problems remain in the handling of multiple modes and mode changes—corresponding to multiple locations and transitions in hybrid automata. Deep symbolic analyses (grouped under the term "structural analysis" in the related community), grounded on solid foundations, are required to generate simulation code. This paper reviews the issues related to multimode DAE systems and proposes algorithms for their analysis. Computer science is instrumental in these works, with a lot to offer to the simulation scientific community.

Keywords: Hybrid automata, multiphysics modeling, DAE, multimode DAE, structural analysis, index reduction

1 Tom Henzinger and Hybrid Systems in computer science

According to the vision of Hybrid Systems by the computer science community [1], a hybrid system possesses *control locations*, characterized by *invariants* expressed as properties on continuous states; in each location, continuous-time dynamics are specified using Ordinary Differential Equations (ODEs); transitions between control locations are characterized by their *pre-* and *post-conditions*

or, alternatively, by pre-conditions and a reset action. For the class of hybrid automata, a large body of knowledge, techniques and tools has been developed since the pioneering paper [1]. Tom Henzinger himself contributed extensively to this body of knowledge, with theory and tools [17]. Interestingly enough, this vision still grounds commercial modeling tools, e.g., Stateflow, which highlights the depth of the 1990's vision of hybrid automata. Still, pushed by practical considerations, alternative approaches have emerged in the 1990's in the control engineering community, with ties going back to *bond graphs* [27], a formalism that grounds the widely used industrial tool Simcenter Amesim.

2 Cyber-Physical Systems modeling: the need for DAEs

As a running example, we consider the idealized clutch of Fig. 1, involving two rotating shafts with no motor or brake being connected. Each shaft possesses its individual dynamics, relating rotational velocity and torque. This clutch possesses two modes: the two shafts evolve freely when the clutch is released, and are coupled by a perfect contact when the clutch is engaged. At the mode change when the clutch gets engaged, a discontinuity occurs in the rotation velocities (they are generally different before engagement, and get equal in zero time as a result of the engagement). This causes an impulse in the torques. This part of the modeling task requires particular care, as we shall see.

2.1 Modeling from first principles of physics naturally leads to considering acausal models [12]

Modeling the clutch with hybrid state machines: Let us first consider the model of the clutch expressed using ODE-based hybrid state machines.

$$\text{released location} \quad \boxed{\begin{array}{l}\omega_1' = f_1(\omega_1, \tau_1)\\ \tau_1 = 0\\ \omega_2' = f_2(\omega_2, \tau_2)\\ \tau_2 = 0\end{array}} \quad \begin{array}{c}\xrightarrow[\text{restart}_{\mathrm{F}\to\mathrm{T}}]{\gamma:\mathrm{F}\to\mathrm{T}}\\ \\ \xleftarrow[\text{restart}_{\mathrm{T}\to\mathrm{F}}]{\gamma:\mathrm{T}\to\mathrm{F}}\end{array} \quad \boxed{\begin{array}{l}\omega' = f_{12}(\omega, \tau)\\ \tau = 0\end{array}} \quad \text{engaged location} \tag{1}$$

State machine (1) possesses two locations: released (left) and engaged (right). In the left-to-right transition, the label "$\gamma : \mathrm{F} \to \mathrm{T}$" sitting above the transition indicates the event that triggers the transition; the label "$\text{restart}_{\mathrm{F}\to\mathrm{T}}$" sitting below the transition refers to the restart action resulting from the transition—its explicit formulation is given below.

The ODE sitting in the released location is the model of two non-interacting shafts; for each one, the angular velocity ω_i and torque τ_i are related via an ODE, with the torque being zero when the clutch is released. The ODE sitting in the engaged location is clear as well: it is the model of a single shaft obtained by gluing the two shafts together; as such, the "engaged" model is global and its architecture does not reflect the physical architecture.

The two restart actions are now detailed. First, when the clutch gets released ($\text{T} \to \text{F}$), the restart values for the two velocities are equal to the velocity of the global shaft just before mode change:

$$\text{restart}_{\text{T}\to\text{F}} \ : \ \omega_1^+ = \omega_2^+ = \omega^- \ . \tag{2}$$

The symmetric restart action ($\text{F} \to \text{T}$) is more involved. To derive it, we need to get closer to the physics, by specializing the two functions f_i for $i = 1, 2$:

$$f_i(\omega_i, \tau_i) = \tfrac{1}{J_i}(\tau_i - g_i(\omega_i)) \ , \tag{3}$$

where J_i is the moment of inertia of shaft i, and $g_i(\omega_i)$ is the friction. When the clutch gets engaged, the two shaft velocities are discontinuous, as they become equal in zero time. The resulting common value follows from the law of preservation of angular momentum, which writes: $(J_1 + J_2)\omega^+ = J_1\omega_1^- + J_2\omega_2^-$. Hence, the restart action when the clutch gets engaged is given by:

$$\text{restart}_{\text{F}\to\text{T}} \ : \ \omega^+ = \frac{J_1\omega_1^- + J_2\omega_2^-}{J_1 + J_2} \ . \tag{4}$$

Formulas (1)–(4) fully specify the model of the clutch using ODE-based hybrid state machines. It can then easily be brought to the classical Alur-Henzinger hybrid automaton form [1]; this transformation has no impact on the issues discussed below.

DAE-based modeling: Generally, modeling in physics does not rely on hybrid automata. Let us consider electrical circuits as an illustration. First, the Kirchhoff laws are expressed as balance equations: the sum of currents at a node is zero; the sum of voltages along a loop is zero. Second, most components (e.g., resistors and capacitors) come with no prespecified input/output orientation. A similar situation actually arises in mechanics or in thermodynamics. What happens if we follow a similar approach for the clutch?

Getting the clutch model from first principles is straightforward and elegant, as shown in Fig. 1. The system and model architectures coincide: the overall model consists of a model of each individual shaft, completed with a model for the junction (the velocities coincide and the torques sum up to zero if the clutch is engaged, $\gamma = \text{T}$; the torques are zero if the clutch is released, $\gamma = \text{F}$). At an instant of mode change when the cluch gets engaged, the *a priori* different velocities of the two shafts will, in zero time, merge to a single identical velocity while the efforts on the shafts will climb up to impulses. The resulting common velocity will depend, in a determinate way, on the inertia and velocity of each shaft prior to engagement. Thus, the main characteristics of the model of Fig. 1 are the following:

1. **The model has two modes:** γ=F and γ=T yield different dynamics.
2. **The model is acausal:** In the released mode γ=F, the two torques are inputs and the two angular positions are outputs; in the engaged mode γ=T, both angular positions and torques are unknowns, with no input/output orientation being defined: the model is acausal in this mode.

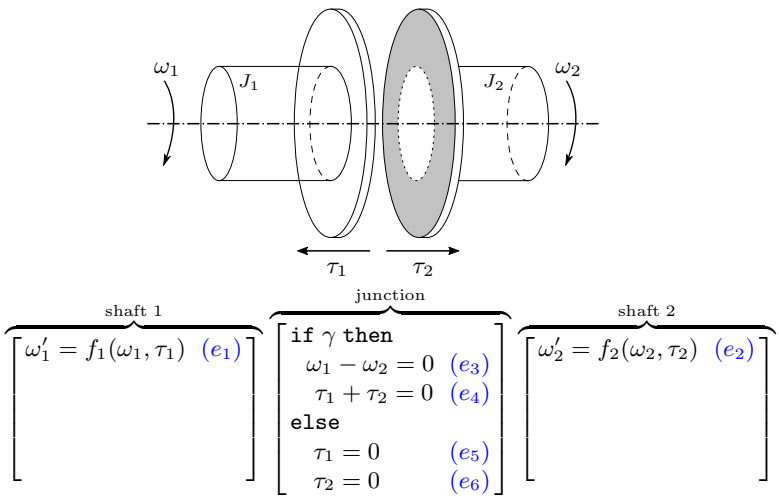


Fig. 1. A simple clutch with two shafts and its model from first principles.

3. **The mode transition is not explicitly specified:** No equation directly addresses mode changes $\gamma : \text{T} \to \text{F}$ and $\gamma : \text{F} \to \text{T}$. Still, restart conditions can be automatically inferred from the model of Fig. 1 at compile time, as we shall see.

Modularity in modeling: What happens if we have a chain of clutches with N shafts? The DAE-based model of Fig. 1 just extends with more replicates. In contrast, the Hybrid Automaton model (1,2,4) must be globally rebuilt: the number of states is 2^{N-1}; the model in each state is modified if we glue one more shaft to the chain; the restart conditions are global and we must compute the equivalent moments of inertia for each chain of shafts with engaged clutches. The conclusion is clear: *physical modeling with hybrid automata is not modular; multimode DAE-based modeling is stringently needed to achieve modularity.*

Conclusion: Acausal modeling makes the specification of large Cyber Physical Systems more modular, comfortable and elegant, and allows for better reusability. As a matter of fact, multimode DAE-based modeling is routinely used in focused areas such as electrical circuits [16], contact mechanics [24], or hydraulics. The modeling of large CPS, however, requires *physics-agnostic modeling* allowing for the combination of different physics together with the embedded control software. The Modelica language, which supports physics-agnostic multimode modeling since version 3.3 [13,4], is a *de facto* standard in this area.

2.2 A new avenue of paradigms, issues, and questions

From the observations above, we identify new paradigms, issues and questions for the computer science community:

- **How can we formally define the class of models that contains the clutch model shown in Fig. 1?**
- **Is there room for a computer scientist's way of thinking in this new area?**

Issue 1 (acausal models) *Rather than only considering ODEs (i.e., equations of the form* $y'=g(y,u)$ *where* y *is the state and* u *is the input),* acausal modeling *relies on DAEs, i.e., equations of the form* $f(x',x,v)=0$ *where both* x *and* v *are unknowns, with no specification of any input/output status. What are the consequences in terms of generating simulation code?*

Issue 1 was addressed by the DAE community by introducing the notion of *index* [10,22]. An abstraction of the notion of index, in which regularity/singularity of the Jacobian is considered in a generic sense, was proposed as a way to better scale up [23,25]. This solution, known as *structural analysis* of DAE systems, is extensively used in Modelica tools.

In addition to being naturally described by an acausal model, a system may have different modes for various reasons. Mechanical impacts where the bodies may remain in contact after a collision or an active grasping/docking, idealized electrical or hydraulic switching elements, are examples of situations requiring different sets of equations for their modeling. Another reason is the control of complex scenarios like repairing a satellite with robots, or switching between start-up, normal operation and shutdown during the operation of a power plant. Models of this kind are called *multimode* models. Hence the second issue for consideration:

Issue 2 (multiple modes) *What is the consequence of considering* multimode DAE models*—in other words,* DAE-based hybrid systems*?*

Issue 2 was identified by authors, but is still incorrectly addressed in industrially used DAE-based modeling tools, as we shall see in the next Section. Authors have proposed dynamic approaches that perform structural analysis when needed [18,19,14], at the cost of significant computational overheads at runtime. In addition, the compile time assistance in model debugging provided by structural analysis is lost with such approaches. On top of that, new difficulties can also arise at mode changes:

Issue 3 (synthesis of restarts at mode changes) *The designer specifies the model for each mode, but should not always be requested to specify the restart conditions resulting from a transition. How can these conditions be inferred?*

For example, restart equations for the clutch model (when entering the engaged mode) should be *synthesized* during compilation, not manually specified. For larger models, restart conditions may be very difficult to synthetize by hand, which makes Issue 3 of uttermost importance.

Last but definitely not least, we lack a reference semantics for multimode DAE systems in general:

Issue 4 (notion of solution) *No notion of solution exists for general multimode DAE systems, but only partial notions on restricted subclasses of systems.*

This came as a surprise to us when we started our study—the reader is referred to [4] for a bibliographical discussion. The work of Stephan Trenn, relying on distributions, is an important contribution to the subject. However, Trenn himself points out in [28,29] inherent difficulties to this approach, which indicates that distributions are not the ultimate answer to deal with impulsive variables in multimode DAE systems. Still, Trenn was able in [20] to define complete solutions for a class of switched DAE systems in which each mode is in *quasi-linear form*: switching conditions are time-based, not state-based. The bottom line is that, unlike for ODE-based hybrid systems or DAE-based single-mode systems, we lack reference semantics. Moreover, spurious situations can arise, leading to infinitely fast mode switching [30]. The best we can hope is to prove that our approach boils down to known approaches for certain subclasses of systems: such an equivalence result was proved in [5] for the subclass of *semi-linear DAE systems* supported by Modiamath [14].

Contribution: In [9], we addressed Issue 2 by proposing a scalable, compile-time, multimode structural analysis; in this paper, we present our approach to Issues 3 and 4. For this purpose, we need to give semantics to dynamics that combine discrete steps and continuous evolutions. This was already an issue for ODE-based hybrid systems; it becomes a steep challenge when moving to DAEs. The use of *nonstandard analysis* and *hyperreals* was already advocated in the study of ODE-based hybrid systems in the computer science community [3,2]. While it was an "alternative approach" in that case, it turned out to become instrumental for the study of DAE-based hybrid systems.

3 The clutch example in existing physical modeling tools

The clutch model of Fig. 1 translates as the Modelica code of Fig. 2, which is accepted by the two Modelica tools we had the opportunity to test: OpenModelica 1.17.0 [15] and Dymola 2021 [11]. However, none of these tools generates correct simulation code. Simulations fail precisely at the instant when the clutch switches from the released mode (`g=false`) to the engaged one (`g=true`), as evidenced by a division by zero exception resulting from the pivoting of a linear system of equations that becomes singular when `g` becomes equal to `true`.

We also tested the Mathematica Advanced Hybrid & DAE toolbox, a library advertised as supporting multimode DAE systems. With this toolbox, the simulation does not abort, but it returns incorrect results: the angular momentum of the system is not preserved, in contradiction with the physics; also, the state jump seems to be randomly selected: a small change in the velocities before the change results in totally different values for the restart.

```
model ClutchBasic
parameter Real w01=1;
parameter Real w02=1.5;
parameter Real j1=1;
parameter Real j2=2;
parameter Real k1=0.01;
parameter Real k2=0.0125;
parameter Real t1=5;
parameter Real t2=7;
Boolean g(start=false);
Real w1(start=w01, fixed=true);
Real w2(start=w02, fixed=true);
Real f1; Real f2;
equation
  g = (time>=t1) and (time<=t2);
  j1*der(w1) = -k1*w1 + f1;
  j2*der(w2) = -k2*w2 + f2;
  0 = if g then w1-w2 else f1;
  f1 + f2 = 0;
end ClutchBasic;
```

Fig. 2. Modelica code for the idealized clutch. This is a faithful translation in the Modelica language of the model of Fig. 1 with f_i as in (3). The input guard γ (called `g`) takes the value `T` between t_1 and t_2, and `F` elsewhere.

Takeaway: Fundamental studies are clearly needed to ensure a correct handling of events of mode change by the Modelica tools. Smoothing "`if then else`" equations could help solving the problems arising at mode changes, but requires a delicate and definitely non-modular tuning, as this tuning depends on the different time scales arising in the system. *We advocate that, as the tools reputedly support multimode DAE models, they should handle them correctly.*

Analysis of the clutch example: In Section 4, we perform the structural analysis of the model in each individual mode. This kind of symbolic analysis for acausal, DAE-based, models provided the grounds for the handling of mode changes that we propose in Section 5. The clutch example illustrates a major difficulty of this task, namely, the fact that restart conditions are, in general, not explicitly specified.

4 Structural analysis of each individual mode

In this section, we analyze separately the model for each mode of the clutch. In the released mode ($\gamma =$ `F` in the model of Fig. 1), the two shafts are independent and one obtains the following two independent ODEs for ω_1 and ω_2:

$$\begin{array}{llll} \omega_1' = f_1(\omega_1, \tau_1) & (e_1) & \tau_1 = 0 & (e_5) \\ \omega_2' = f_2(\omega_2, \tau_2) & (e_2) & \tau_2 = 0 & (e_6) \end{array} \tag{5}$$

In the engaged mode, however ($\gamma =$ `T` in the model of Fig. 1), the two velocities and torques are algebraically related:

$$\begin{array}{llll} \omega_1' = f_1(\omega_1, \tau_1) & (e_1) & \omega_1 - \omega_2 = 0 & (e_3) \\ \omega_2' = f_2(\omega_2, \tau_2) & (e_2) & \tau_1 + \tau_2 = 0 & (e_4) \end{array} \tag{6}$$

Equation (e_3) relates the two velocities ω_1 and ω_2 that are otherwise subject to the ODE system (e_1, e_2). This makes System (6) a DAE instead of an ODE.

In (6), torques τ_1 and τ_2 are not differentiated: they are *algebraic variables*; velocities ω_1 and ω_2 appear together with derivatives: they are *state variables*; $\omega_1', \omega_2', \tau_1, \tau_2$ are the *leading variables* of System (6): they represent the unknowns.

Suppose that the leading variables $\omega_1', \omega_2', \tau_1, \tau_2$ are determined by given *consistent* values for the state variables ω_1, ω_2, i.e., values satisfying (e_3). Then, by using an ODE solver, one could perform an integration step and update the current velocities ω_1, ω_2 using the computed values for their derivatives ω_1', ω_2'. In this case, we say that the considered DAE is an "extended ODE" [25].

It turns out that this condition does not hold for System (6) as is. To intuitively explain the issue, we move to sampled time $\mathbb{T}_\delta =_{\text{def}} \{0, \delta, 2\delta, 3\delta, \dots\}$, by applying an explicit first-order Euler scheme with constant step size $\delta > 0$:

$$\begin{array}{llll} \omega_1^\bullet = \omega_1 + \delta\, f_1(\omega_1, \tau_1) & (e_1^\delta) & \omega_1 - \omega_2 = 0 & (e_3) \\ \omega_2^\bullet = \omega_2 + \delta\, f_2(\omega_2, \tau_2) & (e_2^\delta) & \tau_1 + \tau_2 = 0 & (e_4) \end{array}, \tag{7}$$

where, for every $t \in \mathbb{T}_\delta$,

$$\omega^\bullet(t) =_{\text{def}} \omega(t+\delta) \quad \text{and} \quad {}^\bullet\omega(t) =_{\text{def}} \omega(t-\delta) \tag{8}$$

respectively denote the forward and backward time shift operators by an amount of δ. Suppose we are given consistent values $\omega_1 = \omega_2$ and we wish to use System (7), seen as a system of algebraic equations, to determine the value of the dependent variables (i.e., the unknowns) $\tau_1, \tau_2, \omega_1^\bullet, \omega_2^\bullet$. This attempt fails since we have only three equations e_1^δ, e_2^δ and e_4 to determine four unknowns τ_1, τ_2, $\omega_1^\bullet$ and $\omega_2^\bullet$; indeed, the omitted equation (e_3) does not involve any leading variable.

Since System (7) is time invariant, if the system remains in the engaged mode for at least δ seconds, then the following shifted *latent equation* also holds:

$$\omega_1^\bullet - \omega_2^\bullet = 0 \quad (e_3^\bullet) \tag{9}$$

Replacing (e_3) with $(e_3^\bullet)$ in System (7) yields a system with four equations and four dependent variables, which is nonsingular in a generic sense. One can now use System (7) augmented with equation $(e_3^\bullet)$ to get an execution scheme for the engaged mode of the clutch. This is shown in Execution Scheme 1.

Execution Scheme 1 System (7)+Eq. (9).

Require: consistent ω_1 and ω_2, i.e., satisfying (e_3).

1: `Solve` $\{e_1^\delta, e_2^\delta, e_3^\bullet, e_4\}$ `for` $(\omega_1^\bullet, \omega_2^\bullet, \tau_1, \tau_2)$ ▷ 4 equations, 4 unknowns
2: $(\omega_1, \omega_2) \leftarrow (\omega_1^\bullet, \omega_2^\bullet)$ ▷ update the states (ω_1, ω_2)
3: `Tick` ▷ move to next discrete step

Since the next values of the state variables satisfy (9) by construction, the consistency condition is met at the next iteration step.

Comment 1 (structural nonsingularity [5,6]) The implicit assumption behind Line 1 in Execution Scheme 1 is that solving $\{e_1^\delta, e_2^\delta, e_3^\bullet, e_4\}$ always returns a unique set

of values. In our example, this is true in a "generic" or "structural" sense, meaning that it holds for all but exceptional values for variables and/or parameters.[1] We refer to this as *structural nonsingularity* in the sequel. □

Observe that the same analysis could be applied to the original continuous-time dynamics from System (6) by replacing (e_3) by the differentiated *latent equation*

$$\omega_1' - \omega_2' = 0 \quad (e_3') \tag{10}$$

Since (e_3) holds at any instant, (e_3') follows as long as the solution is smooth enough for the derivatives ω_1' and ω_2' to be defined. The resulting execution scheme is given in Execution Scheme 2, which parallels Execution Scheme 1.

Execution Scheme 2 System (6)+Eq. (10).

Require: consistent ω_1 and ω_2, i.e., satisfying (e_3).

1: `Solve` $\{e_1, e_2, e_3', e_4\}$ `for` $(\omega_1', \omega_2', \tau_1, \tau_2)$ ▷ 4 equations, 4 unknowns
2: `ODESolve` (ω_1, ω_2) ▷ update the states (ω_1, ω_2)
3: `Tick` ▷ move to next discretization step

Line 1 is identical for the two schemes and is assumed to give a unique solution, generically; it fails if one omits the latent equation (e_3'). Then, although getting the next values for ω_1 and ω_2 is easy in Execution Scheme 1, the situation changes in Execution Scheme 2: the derivatives ω_1', ω_2' are first evaluated, then an ODE solver (here denoted by `ODESolve`) is used to update the state (ω_1, ω_2) until the next time horizon.

Comparing Line 1 of Execution Scheme 1 and Line 1 of Execution Scheme 2 reveals the common principle driving the execution of dAE/DAE-based models:[2]

Principle 1 (dAE/DAE execution) *Assume an algebraic equation system solver. The key task is to massage the source model in such a way that the system of equations determining the leading variables becomes regular (i.e., possesses, locally, a unique solution).*

This transforms the original DAE system $F(X', X, Y) = 0$ into a form equivalent to an ODE $X' = G(X, Y)$; the same holds for the discrete-time counterpart.

Differentiation index and structural analysis of DAEs: The replacement of (e_3) by (e_3'), which resulted in Execution Scheme 2, is known in the literature as *index reduction* [22]. It requires adding the (smallest set of) latent equations

[1] Numerical singularity occurs when the Jacobian matrix of the system is singular; in this specific case, this can only occur when $\partial f_1(\omega_1, \tau_1)/\partial\tau_1 + \partial f_2(\omega_2, \tau_2)/\partial\tau_2 = 0$. Unlike structural singularity, numerical singularity depends on numerical values of both coefficients and variables.

[2] dAE stands for *difference* Algebraic Equation (related to discrete time), while DAE stands for *Differential* Algebraic Equation (related to continuous time).

needed for the system in Line 1 of the execution scheme to become solvable and deterministic. The number of successive time shifts or differentiations needed for obtaining all the latent equations is called the *differentiation index* [10]. Generally, for any given DAE model S, which is a system of numerical equations

$$f_i(\text{the } x_j \text{ and their derivatives}) = 0\ , \quad i = 1, \ldots, m;\ \ j = 1, \ldots, n\ , \qquad (11)$$

the structural analysis consists in producing Execution Scheme 2 associated to this model. By the implicit function theorem, the existence and uniqueness of solutions of System (11) rely on the invertibility of the Jacobian matrix collecting the partial derivatives $\partial f_i/\partial u_j$, where u_j is the leading variable associated to the real variable x_j.

An approximate analysis of System (11) was developed since the 1980's that guarantees the regularity of this Jacobian matrix in a structural sense, taking only into account the occurrence/absence of a variable in an equation and its differentiation order: this data is stored into the *weighted incidence graph* $\mathcal{G}_S$ of the system, which is a labeled bipartite graph in which edge (f_i, x_j) is present with weight d_{ij} iff variable x_j occurs in f_i with degree d_{ij} [23,22,25]. In [25], the problem of finding the *offsets* of each equation f_i (i.e., the number of times it should be differentiated) was reduced to solving a certain linear program associated to the weighted incidence graph of System (11); this is known as the *Σ-method*. Details are found in [5,6].

5 Handling mode changes

Caveat: What is the semantics of a multimode DAE? The semantics of ODEs (i.e., the sets of runs or trajectories they define) has been known for a long time, as existence and uniqueness theorems for their solutions are well known in mathematics. The same holds for DAEs, although the corresponding developments are much more recent [10]. In formal studies on hybrid systems, defining the semantics of a model is usually not an issue, hence, the scientific community can focus on difficult decidability issues for various classes of properties and models.

When entering the domain of multimode DAE systems, we did not expect semantics to be a problem. Many physical subdomains involve multimode DAE for their models. Electrical circuits with ideal diodes [16], contact and friction mechanics [24], and hydraulics with multiple-phase materials, are examples of this. For each of these domains, solutions of the corresponding multimode DAEs are well defined, albeit with reference to the dedicated formalism used to describe them—these formalisms differ among the various fields of physics.

In contrast, multiphysics modeling combined with the modeling of embedded digital control is beyond the scope of such dedicated modeling approaches. A notable exception stems from *bond graphs* [27], where generic notions of *effort*, *flow*, *junction*, and *gyrator* generalize the notions of potential, current, junction, and motor in electrical circuits. This yields a generic multiphysics formalism well suited to DAE-based modeling. However, so-called *switch bond graphs*, extending

bond graphs to multimode systems, are not well developed [8]. We are therefore faced with the following demanding problem:

Problem 1 (Semantics). Given that there is no reference semantics in mathematics for general multimode DAE systems, find a sensible subclass of multimode DAE systems:

1. *that is rich enough for supporting complex multiphysics systems modeling,*
2. *so that checking that a given system belongs to this subclass can be checked at compile time, and*
3. *for which a semantics is well defined.*

An example of property that does not meet requirement 2 is "the class of systems whose solution trajectories are continuous with continuous derivatives". Our work [5,6] provides a systematic answer to Problem 1. We illustrate our approach on the clutch example in the next section.

5.1 Structural analysis at mode changes

As discussed above, discontinuous velocities and impulsive torques are expected when the clutch gets engaged. To analyze this, we reuse the discrete-time approximation (7) with time step δ and the forward and backward shift operators introduced in (8). To capture the transition $\gamma : \text{F} \to \text{T}$, we unfold the dynamics of the clutch over two successive instants, by adding the latent equation $(e_3^\bullet)$ associated to mode $\gamma{=}\text{T}$:

$$\begin{array}{r}\text{previous}\\ \text{instant}\\ \gamma=\text{F}\end{array}\left\{\begin{array}{ll}\frac{\omega_1-{}^\bullet\omega_1}{\delta}=f_1({}^\bullet\omega_1,{}^\bullet\tau_1) & ({}^\bullet e_1^\delta)\\ \frac{\omega_2-{}^\bullet\omega_2}{\delta}=f_2({}^\bullet\omega_2,{}^\bullet\tau_2) & ({}^\bullet e_2^\delta)\\ {}^\bullet\tau_1=0 & \\ {}^\bullet\tau_2=0 & \end{array}\right.$$
$$\begin{array}{r}\text{current}\\ \text{instant}\\ \gamma=\text{T}\end{array}\left\{\begin{array}{ll}\frac{\omega_1^\bullet-\omega_1}{\delta}=f_1(\omega_1,\tau_1) & \\ \frac{\omega_2^\bullet-\omega_2}{\delta}=f_2(\omega_2,\tau_2) & \\ \omega_1-\omega_2=0 & (e_3)\\ \omega_1^\bullet-\omega_2^\bullet=0 & (e_3^\bullet)\\ \tau_1+\tau_2=0 & \end{array}\right. \qquad \left(\begin{array}{c}\text{subsystem}\\ ({}^\bullet e_1^\delta,{}^\bullet e_2^\delta,e_3)\\ \text{is conflicting}\end{array}\right) \tag{12}$$

Subsystem $({}^\bullet e_1^\delta,{}^\bullet e_2^\delta,e_3)$ possesses 3 equations, but only 2 dependent variables ω_1,ω_2 as the remaining variables are all determined at the previous time step. This subsystem is thus structurally singular (over-specified). Should the clutch model be rejected? Not quite. This problem is only an artifact of time discretization.

We will get rid of the conflict in model (12) by removing some equations from the conflicting subsystem, at the instant of mode change. To respect causality, the past will not be undone, so that equations $({}^\bullet e_1^\delta,{}^\bullet e_2^\delta)$, which were handled at

the previous instant, cannot be modified. However, *we can resolve the conflict by erasing equation* (e_3) *in System* (12). Thus, at the instant of mode change, the consistency equation (e_3) from the new mode will not be satisfied. At the next instant, however, consistency equation $\omega_1 - \omega_2 = 0$ will be guaranteed by equation $(e_3^\bullet)$ of (12), and it will then remain satisfied as long as the system stays in the engaged mode. Hence, erasing (e_3) in System (12) amounts to postponing by one time step the satisfaction of this state constraint.

Acceptance/rejection of a model: The analysis of the released $\rightarrow$ engaged mode change succeeds for the clutch example. It might have failed if erasing contradicting equations in the present and future of the mode change had yielded a structurally under-specified system, expressing that the model would be under-specified at this mode change—see the cup-and-ball example in [5,6].

Discussion: The analysis developed here meets the requirements of Problem 1: the acceptance/rejection of the submitted model relies on a compile-time structural analysis. For an accepted model, the δ-discretized scheme defines the semantics, which is well defined in a generic sense.

Working with a fixed step size discretization of time is, however, unacceptable, even for single-mode ODEs, as stiff dynamics, whose presence cannot be checked at compile time, can always occur. In the next section, we address this difficulty by taking for our time step an "infinitesimal" number, thus getting "infinitesimal" accuracy for our discretization. This is achieved by calling nonstandard analysis to the rescue.

5.2 Nonstandard analysis and hyperreals to the rescue

If DAE dynamics is approximated in discrete time, then the whole model becomes discrete-time. To avoid the problem of approximation error, our idea is to use an "infinitesimal" time step in the discrete-time approximation. This will yield an approximation that is accurate up to an infinitesimal error. This can be made rigorous by relying on *nonstandard analysis* [26,21,5], which extends the set $\mathbb{R}$ of real numbers to a superset ${}^\star\mathbb{R}$ of *hyperreals* that includes infinitely large and infinitely small numbers. For the understanding of this paper, it is enough to know the following about nonstandard analysis.

A glimpse of nonstandard analysis: There exist ***infinitesimals***, defined as hyperreals of ${}^\star\mathbb{R}$ that are smaller in absolute value than any nonzero real number. The usual arithmetic operations and relations are lifted to ${}^\star\mathbb{R}$. Say that $x \in {}^\star\mathbb{R}$ is *finite* if there exists a real number $y \in \mathbb{R}_+$ such that $|x| \leq y$.

> For every finite hyperreal $x \in {}^\star\mathbb{R}$, there is a unique standard real number $st(x) \in \mathbb{R}$ such that $st(x) - x$ is infinitesimal; $st(x)$ is called the *standard part* (or ***standardization***) of x. (13)

Standardizing functions or systems of equations, however, requires some care. One important issue is derivatives. For $t \mapsto x(t)$ an $\mathbb{R}$-valued signal ($t \in \mathbb{R}$),

$$\begin{array}{l} x \text{ is differentiable at instant } t{\in}\mathbb{R} \text{ if and only if there exists } a{\in}\mathbb{R} \text{ such} \\ \text{that, for any infinitesimal } \varepsilon \in {}^\star\mathbb{R},\ \frac{x(t+\varepsilon)-x(t)}{\varepsilon} - a \text{ is infinitesimal;} \\ \text{then, } a = x'(t). \end{array} \tag{14}$$

Nonstandard time: For $\varepsilon > 0$ an infinitesimal time step, we consider the time index set $\mathbb{T} \subseteq {}^\star\mathbb{R}$:

$$\mathbb{T} = 0, \varepsilon, 2\varepsilon, 3\varepsilon, \cdots = \{n\varepsilon \mid n \in {}^\star\mathbb{N}\} \tag{15}$$

where ${}^\star\mathbb{N}$ denotes the set of *hyperintegers*, consisting of all integers augmented with additional infinite numbers called *nonstandard*. The following features of $\mathbb{T}$ are important: (1) any finite real time $t \in \mathbb{R}$ is infinitesimally close to some element of $\mathbb{T}$ (hence, $\mathbb{T}$ covers $\mathbb{R}$ and can be used to index continuous-time dynamics); and (2) $\mathbb{T}$ is "discrete": every instant $n\varepsilon$ has a predecessor $(n{-}1)\varepsilon$ (except for $n = 0$) and a successor $(n{+}1)\varepsilon$. Let x be a nonstandard signal indexed by $\mathbb{T}$. The *forward-* and *backward-shifted* signals $x^\bullet$ and ${}^\bullet x$ are defined as in (8), with ε replacing δ.

Nonstandard semantics: Solutions of multi-mode DAE systems may be non-differentiable or even non-continuous at events of mode change. To give a meaning to x' at any instant, *we define it everywhere as*

$$x' =_{\text{def}} \frac{1}{\varepsilon}(x^\bullet - x)\,, \tag{16}$$

which agrees with (14) at the instants where x is differentiable. Consider again System (12) where conflicting equation (e_3) is removed, and focus on the dynamics in the current instant, where derivatives are expanded using (16):

$$\begin{cases} \omega_1^\bullet = \omega_1 + \varepsilon\, f_1(\omega_1, \tau_1) \\ \omega_2^\bullet = \omega_2 + \varepsilon\, f_2(\omega_2, \tau_2) \\ \omega_1^\bullet - \omega_2^\bullet = 0 \ \ \texttt{and} \ \ \tau_1 + \tau_2 = 0 \end{cases} \tag{17}$$

We wish to use System (17) by identifying current values for the states ω_i with the *left-limits* ω_i^- i.e., the values of the velocities just before the mode change. From these values, we would then compute the restart values for the velocities $\omega_i^+ =_{\text{def}} \omega_i^\bullet$, together with the torques τ_i.

5.3 From hyperreals to effective code

Unfortunately, hyperreals are unknown to computers; hence, System (17) cannot be used as such, but needs to be *standardized,* by "washing out" the infinitesimal time step ε. As it is infinitesimal, it is tempting to get rid of of it in (17) by simply setting $\varepsilon = 0$. Unfortunately, doing this leaves us with a structurally

singular system, since the two torques are then involved in only one equation.[3] This problem of structural singularity is in fact due to the existence of impulsive variables. To discover them in a systematic way, we perform an *impulse analysis*.

Impulse analysis: When the clutch is released, we have $\omega_1 - \omega_2 \neq 0$, generically. Since $\omega_1^\bullet - \omega_2^\bullet = 0$ holds, $\frac{(\omega_1^\bullet - \omega_2^\bullet) - (\omega_1 - \omega_2)}{\varepsilon} = f_1(\omega_1, \tau_1) - f_2(\omega_2, \tau_2)$ cannot be finite because, if it was, then the function $\omega_1 - \omega_2$ would be continuous, contradicting the assumption that $\omega_1 - \omega_2 \neq 0$. Hence, the hyperreal $f_1(\omega_1, \tau_1) - f_2(\omega_2, \tau_2)$ is necessarily infinite. However, we assumed continuous functions f_i and finite state (ω_1, ω_2). Thus, one of the torques τ_i must be infinite at mode change, and because of equation $(e_4) : \tau_1 + \tau_2 = 0$, both torques are in fact infinite, i.e., are *impulsive*.

Eliminating impulsive variables: We now assume that each f_i has the form (3). Erasing (e_3) from the black part of (12) yields the following system of equations, to be solved for $\omega_1^\bullet, \omega_2^\bullet, \tau_1, \tau_2$ at the instant when γ switches from F to T:

$$\begin{cases} J_1\omega_1^\bullet = J_1\omega_1 + \varepsilon(\tau_1 - g_1(\omega_1)) & (e_1^\varepsilon) \\ J_2\omega_2^\bullet = J_2\omega_2 + \varepsilon(\tau_2 - g_2(\omega_2)) & (e_2^\varepsilon) \\ \omega_1^\bullet - \omega_2^\bullet = 0 & (e_3^\bullet) \\ \tau_1 + \tau_2 = 0 & (e_4) \end{cases} \tag{18}$$

We eliminate the impulsive variables from System (18), namely, the two torques. Using (e_4) yields $-\tau_2 = \tau_1 =_{\text{def}} \tau$. Using $(e_3^\bullet)$ by setting $\omega^\bullet =_{\text{def}} \omega_1^\bullet = \omega_2^\bullet$ and adding the two equations (e_1^ε) and (e_2^ε) yields

$$(J_1 + J_2)\omega^\bullet = J_1\omega_1 + J_2\omega_2 - \varepsilon(g_1(\omega_1) + g_1(\omega_2)) \ .$$

It is now legitimate to set ε to 0 to standardize the right-hand side of this equation. Finally, identifying $st(\omega_i) = \omega_i^-$ and $st(\omega_i^\bullet) = \omega_i^+$ synthesizes formula (4). Alternative approaches for the computation of the reset values, which do not require the elimination of impulsive variables, are presented in [5,6].

We thus recovered the restart condition derived from conservation laws of the mechanics in Section 2.1, *by only using physics-agnostic symbolic reasoning*. This addresses Issue 3 of Section 2.2, and can be made fully automatic [5,6].

6 Discussion and perspectives

A new modeling paradigm: In this paper, we advocate the use of multimode DAE systems (or DAE-based hybrid systems) for physical modeling and, thus, CPS modeling. We argued for its stringent need to bring modularity and reuse in modeling. We illustrated on a toy example the difficulties in compiling such models to derive simulation code, and proposed effective solutions. Major lessons from this study are the following:

[3] Actually, this is an example of erroneous standardization of systems of equations. In [5,6] it is proved that standardizing a system of equations by setting $\varepsilon = 0$ is correct if and only if doing so yields a structurally regular system of equations.

– Since DAEs, not ODEs, are involved, we must rely on structural analysis for both all modes and mode changes of the system.
– In many cases, restart conditions at mode changes can be synthesized from a proper structural analysis.
– The compilation steps detailed above provide a compile-time analysis for accepting or rejecting models, based on over/under-specifications of it, within modes or at mode changes.
– An *impulse analysis* is also presented in [5]; this is a compile-time symbolic analysis identifying impulsive variables, which allows for the handling of models involving impulsive behaviors.
– Hyperreals and nonstandard analysis were instrumental in doing this: they allow to unify the model of time, combining continuous-time dynamics within long modes and discrete-time steps for computing restarts at mode changes.

Our approach gives a semantics to multimode DAE systems possessing long modes and finite cascades of mode switches. Note that dense infinite cascades, such as those resulting from sliding mode control [30], are not covered. Our semantics was proved to coincide with the semantics provided by the physics for the restricted class of *semilinear* systems, to which the clutch belongs [4,5]. No comparison can be stated for our method in full generality, however, due to the lack of reference semantics for multimode multiphysics systems. A full justification of the use of nonstandard analysis and standardization was provided in [5].

Making our approach systematic with algorithms that properly scale up: In the grounding paper [5], we develop a comprehensive theory handling, in full generality, multimode DAE systems belonging to the class described above. The whole procedure is automatic, and effective algorithms are specified and mathematically justified.

In this paper, we did not discuss algorithmic complexity. For the clutch example, structural analyses were needed for each of the two modes (released and engaged), and each of the two transitions (released $\rightarrow$ engaged and vice-versa). Since the number of modes can grow exponentially with the number of subsystems, the *mode-centric* modeling approach associating a DAE model to each different mode cannot scale up. This can be overcome in practice by adopting the dual *equation-centric* approach, in which each equation is labeled with a predicate characterizing the set of modes in which this equation is active.[4] This dual approach, combined with an efficient encoding of the multimode Σ-method using Binary Decision Diagrams (BDD) techniques, proved to scale up nicely on significant experiments [9]. A large part of this body is implemented in our tool IsamDAE.[5] The examples coming with this tool already include thermodynamical, electrical and pneumatic models. Work is in progress toward developing

[4] This equation-centric viewpoint was first used in the *clock calculus* and *conditional dependency graph* used in the compilation of the Signal synchronous language [7].

[5] To date, the following compilation tasks are implemented: multimode Σ-method, multimode initialization, and multimode Dulmage-Mendelsohn decomposition.

modular structural analysis as a means to further scale up, and possibly achieve separate compilation of such models.

Grand challenges: Let us wrap up this paper by proposing a few grand challenges for computer scientists in this area.

Correct compilation of multimode DAE systems: With the current reference tools, physically correct models need to be nontrivially twisted and massaged to get simulated, which can require assistance by tool experts. Our work addresses this challenge.

Scaling up: Large industrial models can involve up to hundreds of thousands of equations; examples are found in the energy sector. Such models are not fully supported by the existing tools. Progress is stringently needed on this issue, and should involve both high performance computing and compile-time analyses such as the ones developed above.

Modularity and separate compilation: The first step of model compilation in current tools is a flattening of the model structure. This can yield prohibitive computational costs, particularly for large systems built from many instantiations of a few component classes. As such, separate compilation of components is stringently needed.

Initialization of multimode DAE models: Specifying correct initial conditions is a widely acknowledged difficulty for designers of large DAE-based models, especially for CPS. Significant work was devoted to initialization for (single-mode) DAEs, but this remains a mostly open issue for multimode DAE models.

General concept of solutions for multimode DAEs: This is a core challenge from a mathematical viewpoint, that we partly addressed.

This work was supported by the FUI ModeliScale DOS0066450/00 French national grant (2018-2021) and the Inria IPL ModeliScale large scale initiative (2017-2021, https://team.inria.fr/modeliscale/).

References

1. R. Alur, C. Courcoubetis, T. A. Henzinger, and P. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 209–229. Springer, 1992.
2. A. Benveniste, T. Bourke, B. Caillaud, J. Colaço, C. Pasteur, and M. Pouzet. Building a hybrid systems modeler on synchronous languages principles. *Proc. IEEE*, 106(9):1568–1592, 2018.
3. A. Benveniste, T. Bourke, B. Caillaud, and M. Pouzet. Nonstandard semantics of hybrid systems modelers. *J. Comput. Syst. Sci.*, 78(3):877–910, 2012.
4. A. Benveniste, B. Caillaud, H. Elmqvist, K. Ghorbal, M. Otter, and M. Pouzet. Multi-mode DAE models - challenges, theory and implementation. In B. Steffen and G. J. Woeginger, editors, *Computing and Software Science - State of the Art*

and Perspectives, volume 10000 of *Lecture Notes in Computer Science*, pages 283–310. Springer, 2019.
5. A. Benveniste, B. Caillaud, and M. Malandain. The mathematical foundations of physical systems modeling languages. *Annual Reviews in Control*, 50:72–118, 2020.
6. A. Benveniste, B. Caillaud, and M. Malandain. Structural analysis of multimode DAE systems: summary of results. *CoRR*, abs/2101.05702, 2021.
7. A. Benveniste, P. Caspi, S. A. Edwards, N. Halbwachs, P. Le Guernic, and R. de Simone. The synchronous languages 12 years later. *Proceedings of the IEEE*, 91(1):64–83, 2003.
8. J. Broenink and K. Wijbrans. Describing discontinuities in bond graphs. In *Proc. of the 1st Int. Conf. on bond graph modeling*, volume 25 No 2 of *SCS Simulation Series*, 1993.
9. B. Caillaud, M. Malandain, and J. Thibault. Implicit structural analysis of multimode DAE systems. In A. D. Ames, S. A. Seshia, and J. Deshmukh, editors, *HSCC '20: 23rd ACM International Conference on Hybrid Systems: Computation and Control, Sydney, New South Wales, Australia, April 21-24, 2020*, pages 20:1–20:11. ACM, 2020.
10. S. L. Campbell and C. W. Gear. The index of general nonlinear DAEs. *Numer. Math.*, 72:173–196, 1995.
11. Dassault Systèmes AB. Dymola official webpage. https://www.3ds.com/products-services/catia/products/dymola/, Accessed: 2022-06-01.
12. H. Elmqvist. A structured model language for large continuous systems, 1978. PhD, Lund University.
13. H. Elmqvist, S.-E. Mattsson, and M. Otter. Modelica extensions for multi-mode DAE systems. In H. Tummescheit and K.-E. Arzèn, editors, *Proc. of the 10th Int. Modelica Conference*, Lund, Sweden, Sept. 2014. Modelica Association.
14. H. Elmqvist and M. Otter. Modiamath webpage. https://modiasim.github.io/ModiaMath.jl/stable/index.html, Accessed: 2022-06-01.
15. P. Fritzson, A. Pop, K. Abdelhak, A. Ashgar, B. Bachmann, W. Braun, D. Bouskela, R. Braun, L. Buffoni, F. Casella, R. Castro, R. Franke, D. Fritzson, M. Gebremedhin, A. Heuermann, B. Lie, A. Mengist, L. Mikelsons, K. Moudgalya, L. Ochel, A. Palanisamy, V. Ruge, W. Schamai, M. Sjölund, B. Thiele, J. Tinnerholm, and P. Östlund. The OpenModelica Integrated Environment for Modeling, Simulation, and Model-Based Development. *Modeling, Identification and Control*, 41(4):241–295, 2020.
16. W. Heemels, M. Camlibel, and J. Schumacher. On the dynamic analysis of piecewise-linear networks. *IEEE Transactions on Circuits and Systems I-regular Papers*, 49:315–327, 2002.
17. T. A. Henzinger and P. Ho. HYTECH: The Cornell HYbrid TECHnology Tool. In P. J. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors, *Hybrid Systems II, Proceedings of the Third International Workshop on Hybrid Systems, Ithaca, NY, USA, October 1994*, volume 999 of *Lecture Notes in Computer Science*, pages 265–293. Springer, 1994.
18. C. Höger. Dynamic structural analysis for DAEs. In *Proceedings of the 2014 Summer Simulation Multiconference, SummerSim 2014, Monterey, CA, USA, July 6-10, 2014*, page 12, 2014.
19. C. Höger. Elaborate control: variable-structure modeling from an operational perspective. In *Proceedings of the 8th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools, EOOLT '17, Weßling, Germany, December 1, 2017*, pages 51–60, 2017.

20. D. Liberzon and S. Trenn. Switched nonlinear differential algebraic equations: Solution theory, lyapunov functions, and stability. *Automatica*, 48(5):954–963, 2012.
21. T. Lindstrøm. An invitation to nonstandard analysis. In N. Cutland, editor, *Nonstandard Analysis and its Applications*, pages 1–105. Cambridge Univ. Press, 1988.
22. S. E. Mattsson and G. Söderlind. Index reduction in differential-algebraic equations using dummy derivatives. *Siam J. Sci. Comput.*, 14(3):677–692, 1993.
23. C. Pantelides. The consistent initialization of differential-algebraic systems. *SIAM J. Sci. Stat. Comput.*, 9(2):213–231, 1988.
24. F. Pfeiffer and C. Glocker. *Multibody Dynamics with Unilateral Contacts*. Wiley, 2008.
25. J. D. Pryce. A simple structural analysis method for DAEs. *BIT*, 41(2):364–394, 2001.
26. A. Robinson. *Nonstandard Analysis*. Princeton Landmarks in Mathematics, 1996. ISBN 0-691-04490-2.
27. J. Thoma. *Introduction to Bond Graphs and Their Applications*. Pergamon international library of Science, technology, engineering and social studies. Pergamon Press, 1975.
28. S. Trenn. *Distributional Differential Algebraic Equations*. PhD thesis, Technischen Universität Ilmenau, 2009.
29. S. Trenn. Regularity of distributional differential algebraic equations. *MCSS*, 21(3):229–264, 2009.
30. V. Utkin. Sliding mode control in mechanical systems. In *Industrial Electronics, Control and Instrumentation, 1994. IECON '94., 20th International Conference on*, volume 3, pages 1429 –1431 vol.3, Sept. 1994.