The Mathematical Foundations of Physical Systems Modeling Languages

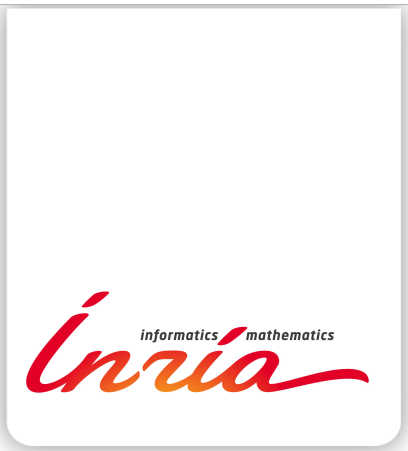
Albert Benveniste, Benoît Caillaud, Mathias Malandain

The Mathematical Foundations of Physical Systems Modeling Languages

Albert Benveniste, Benoit Caillaud, Mathias Malandain

RESEARCH REPORT
N° 9334
August 2020
Project-Team Hycomes

ISSN 0249-6399 ISRN INRIA/RR--9334--FR+ENG



The Mathematical Foundations of Physical Systems Modeling Languages

Albert Benveniste*, Benoit Caillaud, Mathias Malandain

Project-Team Hycomes

Research Report n° 9334 — August 2020 — 112 pages

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/).

* Inria-Rennes, Campus de Beaulieu, 35042 Rennes cedex, France; email: surname.name@inria.fr

RESEARCH CENTRE
RENNES – BRETAGNE ATLANTIQUE

Campus universitaire de Beaulieu
35042 Rennes Cedex

Abstract: Modern modeling languages for general physical systems, such as Modelica, Amesim, or Simscape, rely on Differential Algebraic Equations (DAEs), i.e., constraints of the form $f(x', x, u) = 0$. This drastically facilitates modeling from first principles of the physics, as well as the reuse of models. In this paper, we develop the mathematical theory needed to establish the development of compilers and tools for DAE-based physical modeling languages on solid mathematical bases.
Unlike Ordinary Differential Equations (ODEs, of the form $x' = g(x, u)$), DAEs exhibit subtle issues because of the notion of *differentiation index* and related *latent equations*—ODEs are DAEs of index zero, for which no latent equation needs to be considered. Prior to generating execution code and calling solvers, the compilation of such languages requires a nontrivial *structural analysis* step that reduces the differentiation index to a level acceptable by DAE solvers.
The models supported by tools of the Modelica class involve multiple modes, with mode-dependent DAE-based dynamics and state-dependent mode switching. However, multimode DAEs are much more difficult to handle than DAEs, especially because of the events of mode change. Unfortunately, the large literature devoted to the mathematical analysis of DAEs does not cover the multimode case, typically saying nothing about mode changes. This lack of foundations causes numerous difficulties to the existing modeling tools. Some models are well handled, others are not, with no clear boundary between the two classes. In this paper, we develop a comprehensive mathematical approach supporting compilation and code generation for this class of languages. Its core is the *structural analysis of multimode DAE systems*. As a byproduct of this structural analysis, we propose sound criteria for accepting or rejecting multimode models. Our mathematical development relies on *nonstandard analysis*, which allows us to cast hybrid system dynamics to discrete-time dynamics with infinitesimal step size, thus providing a uniform framework for handling both continuous dynamics and mode change events.

Key-words: structural analysis, differential-algebraic equations (DAE), multi-mode systems, variable-structure models, nonstandard analysis

Fondements mathématiques des langages de modélisation de systèmes physiques

Résumé : Les langages modernes de modélisation de systèmes physiques, tels que Modelica, Amesim, ou Simscape, s'appuient sur des Équations Algébro-Différentielles (DAE), qui sont des contraintes de la forme $f(x', x, u) = 0$. Cette approche rend naturelle la modélisation directe à partir des principes de la physique, ainsi que la réutilisation de modèles. Dans cet article, nous développons les bases mathématiques qui fondent ces langages.

Par rapport aux Équations Différentielles Ordinaires (ODE, de la forme $x' = g(x, u)$), les DAE sont plus compliquées, en raison des notions d'*index* et d'*équations latentes*—les ODE sont des DAE d'index zéro, sans équations latentes. Avant toute génération de code, une *analyse structurelle* des modèles est requise, afin de ramener l'index à des valeurs acceptables pour les solveurs de DAE (deux ou trois maximum).

Les modèles acceptés par les langages de la classe Modelica sont des DAE *multi-modes*, avec une DAE différente dans chaque mode, tandis que les changements de mode sont provoqués par des conditions portant sur l'état et/ou le temps. Les DAE multi-mode sont beaucoup plus difficiles à traiter que les DAE (mono-mode). La difficulté principale est le traitement des changements de mode et des conditions de redémarrage associées. Ce point n'est pas abordé dans la littérature mathématique sur les DAE multi-mode: rien n'est dit sur comment les changements de mode doivent être traités. Cette situation est source de problèmes importants pour tous les outils existants. Certains modèles sont correctement traités, d'autres non, sans qu'on sache bien pourquoi—et les cas problématiques ne sont pas pathologiques, mais se rencontrent naturellement.

Dans ce travail, nous développons une approche cohérente et complète pour la compilation des DAE multi-mode. Au cœur de celle-ci se trouve une *analyse structurelle multi-mode,* qui couvre à la fois les dynamiques dans les divers modes, et les changements de mode. Notre approche traite des modèles présentant des valeurs impulsives pour certaines variables, lors des changements de mode. Notre approche permet d'accepter ou de rejeter des modèles sur des critères bien fondés (modèle sur- ou sous-déterminé). Ces travaux eussent été impossibles sans le recours à l'*analyse non-standard.* Grâce à elle, nous pouvons réinterpréter les dynamiques en temps continu comme du temps discret à pas infinitésimal, ce qui fournit une vision uniforme de toute la trajectoire—en mode continu ou en changement de mode.

Mots-clés : analyse structurelle, équations algébro-différentielles (DAE), systèmes multi-mode, modèles à structure variable, analyse non-standard

Contents

1 Introduction

1.1 Overall motivations

Multimode DAE systems constitute the mathematical framework supporting the physical modeling of systems possessing different *modes*. Each mode exhibits a different dynamics, captured by *Differential Algebraic Equations* (DAEs). Multimode DAE systems are the underlying framework of 'object-oriented' modeling languages such as Modelica, Amesim, or Simscape. Corresponding models can be represented as systems of guarded equations of the form

$$\text{if } \gamma_j(x_i\text{'s and derivatives}) \text{ then } f_j(x_i\text{'s and derivatives}) = 0 \tag{1}$$

where $x_i, i = 1, \ldots, n$ denote the system variables, and, for $j = 1, \ldots, m$, $\gamma_j(\ldots)$ is a Boolean combination of predicates guarding the differential or algebraic equation $f_j(\ldots) = 0$. The meaning is that, if γ_j has the value T (the constant true), then equation $f_j(\ldots) = 0$ has to hold; otherwise, it is discarded. In particular, when all the predicates have the value T, one obtains a *single-mode* DAE, that is, a classical DAE defined by the set of equations

$$f_j(x_i\text{'s and derivatives}) = 0 \tag{2}$$

where $i{=}1, \ldots, n$ and $j{=}1, \ldots, m$. When all f_j's have the special form $x'_j - g_j(x_1, \ldots, x_n)$, one recovers the Ordinary Differential Equations (ODE) $x'_j{=}g_j(x_1, \ldots, x_n)$. DAEs are a strict generalization of ODEs, where the so-called *state variables* $x_1, \ldots, x_n$ are implicitly related to their time derivatives $x'_1, \ldots, x'_n$. This modeling framework is fully compositional, since systems of systems of equations of the form (1) are just systems of equations of the form (1), with no restriction.

DAE systems (having a single mode) are well understood. With comparison to ODE systems, a new difficulty arises with the notion of *differentiation index*, introduced in the late 1980's [42, 21]. For simplicity, in this introduction, we discuss it for the particular case in which System (2) involves all derivatives x'_i but no higher-order derivative. The problem addressed by this notion of differentiation index relates to the Jacobian matrix $\mathbf{J}$ associated to System (2), defined by $\mathbf{J}_{ij} = \partial f_j / \partial x'_i$. If this Jacobian matrix is invertible (requiring, in particular, $m{=}n$), then all the derivatives x'_i are uniquely determined as functions of the x_i, so that System (2) is equivalent to an ODE system. If, however, matrix $\mathbf{J}$ is singular, then System (2) as such is no longer equivalent to an ODE system. This situation can occur, even for a square DAE system possessing a unique solution for any given initial conditions. This situation makes it difficult to design DAE solvers that work for any kind of DAE.

Now, $f_j{=}0$ in (2) implies $\frac{d}{dt} f_j{=}0$, revealing that such *latent equations* come implicitly with System (2). Adding latent equations to the DAE system does not change the system solutions, but it can bring additional constraints on highest-order derivatives of the original system states—on the other hand, this can introduce "spurious" derivatives (of higher order than in the original system), which are to be eliminated. Adding more latent equations while eliminating spurious derivatives eventually leads to highest-order derivatives of the original system states being uniquely determined as functions of the lower order state derivatives. The *differentiation index* [42, 21] is the smallest integer k such that it is enough to differentiate with respect to time every equation $f_j{=}0$ up to order at most k for the above situation to occur.

DAE solvers exist for DAE of index 1, 2, or 3 [37]. An efficient and popular approach, called the *dummy derivatives* method [37], consists in reducing the index of the DAE system to 1, producing in addition a form in which the algebraic constraints are all preserved by the solver—this no longer holds if the index is reduced to zero, leading to an ODE. *Structural analysis* methods have been proposed [40, 48] which perform index reduction by only exploiting the bipartite graph associated to the different equations and variables of the system. Structural analyses scale up much better and provide valid results outside exceptional values for the coefficients arising in the system equations [21]; structural analysis with dummy derivatives is implemented in most Modelica tools. Efforts are still ongoing to improve the efficiency and range of applicability of these methods, but one can say that DAE systems are now reasonably well understood.

In contrast, the handling of mode changes is much less mature. No structural analysis exists for mode changes. Due to discontinuities in system trajectories, the notion of differentiation index does not apply to multimode DAE systems. The notion of solutions of such systems is not even well understood, except for some subclasses of models, such as *semi-linear systems* [8]. Still, modeling languages exist that support multimode DAE systems, e.g., Modelica, VHDL-AMS, and the proprietary languages Amesim and Simscape. However, the lack of mathematical understanding of mode changes can result in a spurious handling of some physically meaningful models. Indeed, with the exception of the recent work [8], the class of "safe" models (well supported by the considered tool) is never characterized, thus leading to a "try-and-see" methodology. This situation motivates our work.

Establishing the mathematical foundations for compilers[1] of multimode DAE systems raises non-classical difficulties. To substantiate this claim, let us compare the following three subject matters:

1. Developing formal verification for a given class of hybrid systems [2, 45];
2. Developing existence/uniqueness results and discretization schemes, for a given class of (single-mode or multimode) DAE systems [3];
3. Developing the mathematical foundations for multiphysics/multimode DAE systems modeling languages (our focus).

For subject 1, restrictions on the class are stated, under which the proposed algorithms are proved correct, and their complexity is analyzed. Similarly, for subject 2, assumptions are stated on which discretization schemes are proved correct, and convergence rates can be given. In both cases, assumptions are formulated and it is the responsibility of the user to check their validity when performing verification or applying discretization schemes.

In contrast, for subject 3, one cannot expect a compiler to check the existence/uniqueness of solutions of a given multimode DAE system. Worse, we cannot expect the user to check this as part of his/her model design activity. In fact, the compiler must handle *any* submitted model, and it has the responsibility for accepting or rejecting a model on the sole basis of syntactic or symbolic (but never numerical) analyses. This is a demanding task that we claim is not well addressed today.

1.2 Some illustration examples

To illustrate the above discussion, we now review two specific examples of multimode DAE systems.

1.2.1 An ideal clutch

This clutch is depicted in Figure 1. It is a simple, idealized clutch involving two rotating shafts where no motor or brake are connected. More precisely, we assume that this system is closed, with no interaction other than explicitly specified.

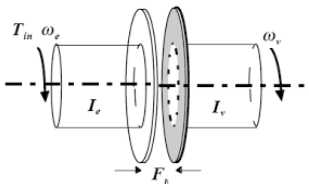


Figure 1: An ideal clutch with two shafts.

We provide hereafter in (3) a model for it, complying with the general form (1). The dynamics of each shaft i is modeled by $\omega_i' = f_i(\omega_i, \tau_i)$ for some, yet unspecified, function f_i, where ω_i is

[1] By *compilation*, we mean here the suite of symbolic analyses and transformations that must be performed prior to generating simulation code.

the angular velocity, τ_i is the torque applied to shaft i, and ω_i' denotes the time derivative of ω_i. Depending on the value of the input Boolean variable γ, the clutch is either engaged ($\gamma = \text{T}$, the constant "true") or released ($\gamma = \text{F}$, the constant "false"). When the clutch is released, the two shafts rotate freely: no torque is applied to them ($\tau_i = 0$). When the clutch is engaged, it ensures a perfect join between the two shafts, forcing them to have the same angular velocity ($\omega_1 - \omega_2 = 0$) and opposite torques ($\tau_1 + \tau_2 = 0$). Here is the model:

$$\left\{\begin{array}{lll} & & \omega_1' = f_1(\omega_1, \tau_1) \quad (e_1) \\ & & \omega_2' = f_2(\omega_2, \tau_2) \quad (e_2) \\ \texttt{if } \gamma & \texttt{then} & \omega_1 - \omega_2 = 0 \quad (e_3) \\ & \texttt{and} & \tau_1 + \tau_2 = 0 \quad (e_4) \\ \texttt{if not } \gamma & \texttt{then} & \tau_1 = 0 \quad (e_5) \\ & \texttt{and} & \tau_2 = 0 \quad (e_6) \end{array}\right. \tag{3}$$

When $\gamma = \text{T}$, equations (e_3, e_4) are active and equations (e_5, e_6) are disabled, and vice-versa when $\gamma = \text{F}$. If the clutch is initially released, then, at the instant of contact, the relative speed of the two rotating shafts jumps to zero; as a consequence, an impulse is expected on the torques. In addition, the model yields an ODE system when the clutch is released, and a DAE system of index 1 when the clutch is engaged, due to equation (e_3) being active in this mode.

Note that the condition $\tau_1 + \tau_2 = 0$ is an invariant of the system in both modes, which reflects the preservation of the angular momentum due to the assumption that the system is closed. Model (3) is one possible specification. One could have instead put $\tau_1 + \tau_2 = 0$ outside the scope of any guard, thus making explicit that this is an invariant—this is actually the form adopted for the Modelica model of Figure 2; then, mode $\gamma = \text{F}$ only states $\tau_1 = 0$. Both models are equivalent. It turns out that our approach yields the same generated code for both source codes.

```
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;
  Real t(start=0, fixed=true);
  Boolean g(start=false);
  Real w1(start = w01, fixed=true);
  Real w2(start= w02, fixed=true);
  Real f1; Real f2;
equation
  der(t) = 1;
  g = (t >= t1) and (t <= 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;
```

Figure 2: Modelica code for the idealized clutch.

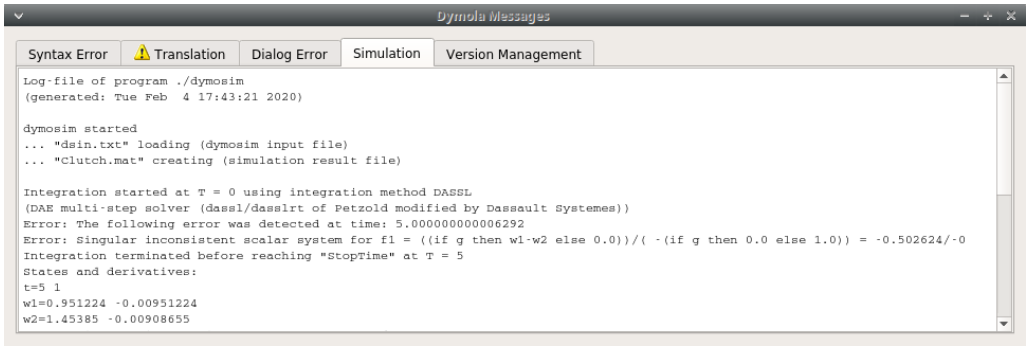


Figure 3: Division by zero exception occuring when simulating the Ideal Clutch Modelica model.

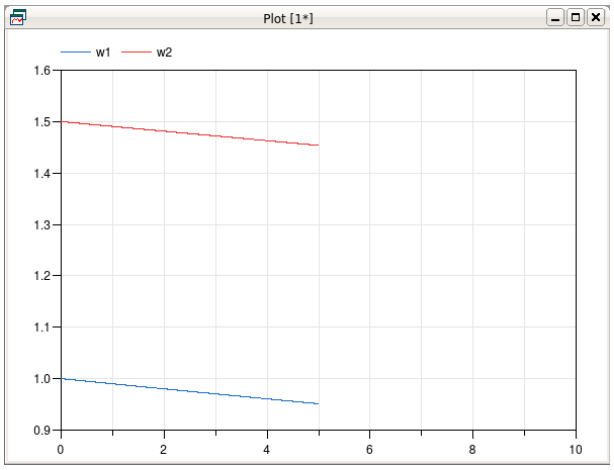


Figure 4: Trajectory of the Ideal Clutch Modelica model: it stops at $t = 5s$, when the exception occurs.

The clutch in Modelica Figure 2 details the Modelica model of the Ideal Clutch system, with two shafts interconnected by an idealized clutch. It is a faithful translation in the Modelica language of the mDAE (3), with the exception that the two differential equations have been linearized. Also, the trajectory of the input guard γ (here called `g`) has been fully defined: it takes the value `T` between instants t_1 and t_2, and `F` elsewhere. The model is deemed to be structurally nonsingular by the two Modelica tools we had the opportunity to test: OpenModelica and Dymola. However, none of these tools generates correct simulation code from this model. Indeed, simulations fail precisely at the instant when the clutch switches from the uncoupled mode (`g=false`) to the coupled one (`g=true`). This is evidenced by a division by zero exception, as shown in Figures 3 and 4.

The root cause of this exception is that none of these tools performs a multimode structural analysis. Instead, the structure of the model is assumed to be invariant, and the structural analysis implemented in these tools is a Dummy Derivatives method, which is proved to be correct only on single-mode DAE systems; it is possibly correct on multimode systems, but under the stringent assumption that the model structure (including, but not limited to, its differentiation index) is independent of the mode. The consequence is that the structural analysis methods implemented in these tools do not detect that the differentiation index jumps from 0 to 1 when the shafts are coupled, and that the structure is not invariant. The division by zero results from the pivoting of a linear system of equations that becomes singular when `g` becomes equal to `true`.

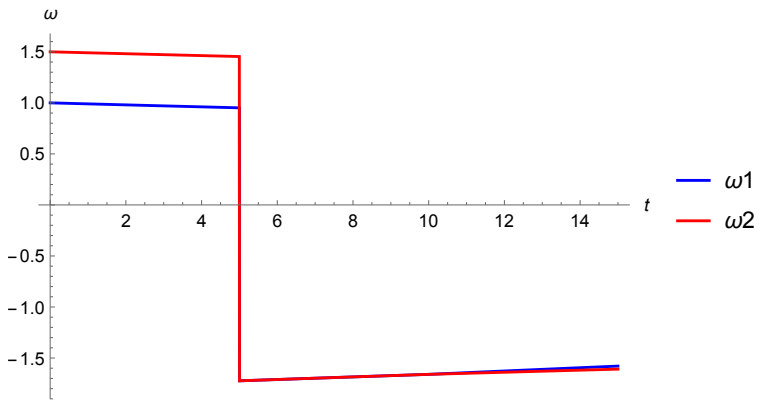


Figure 5: Solution of the Ideal Clutch model, computed by Mathematica.

The clutch in Mathematica DAE systems can be solved with the Mathematica "Advanced Hybrid & DAE" toolbox.[2] This library is also advertised as supporting multimode DAE systems. Figure 5 shows the solution of the Ideal Clutch system, computed by Mathematica. It can be seen that the angular momentum of the mechanical system is not preserved, in contradiction with the physics. Also, a small change in the velocities before the change results in totally different values for restart, although the system is not chaotic by itself. A likely explanation is that Mathematica regards the mode switching as a consistent initialization problem, with one degree of freedom. Indeed, any choice of angular velocities satisfying the constraint $\omega_1 = \omega_2$ is a consistent initial state for the DAE in the engaged mode.

The conclusion of these two experiments is that, clearly, some fundamental study is needed to ensure a correct handling of events of mode change by the Modelica tools. Smoothing the '`if then else`' equation could help solving the above problem, but requires a delicate and definitely non-modular tuning, as it depends on the different time scales arising in the system. We believe that, as the tools reputedly support multimode DAE models, they should handle them correctly.

Our objective For this example, *our objective is to be able to compile System* (3) *and produce correct simulations for it, without any need for smoothing the mode change.* This objective is addressed in Section 2.

1.2.2 A Cup-and-Ball game

Figure 6: The Cup-and-Ball game.

We sketch here a multimode extension of the popular example of the pendulum in Cartesian coordinates [40], namely the planar Cup-and-Ball game, whose 3D version is illustrated by Figure 6. A ball, modeled by a point mass, is attached to one end of a rope, while the other end of the rope is fixed, to the origin of the plane in the model. The ball is subject to the unilateral constraint set by the rope, but moves freely while the distance between the ball and the origin is less than its actual length. Again, we assume that the system is closed and subject to no interaction other than specified. Using Cartesian coordinates x and y, the equations of the planar Cup-and-Ball game are the following:

$$\begin{cases} 0 = x'' + \lambda x & (e_1) \\ 0 = y'' + \lambda y + g & (e_2) \\ 0 \leq L^2 - (x^2+y^2) & (\kappa_1) \\ 0 \leq \lambda & (\kappa_2) \\ 0 = \left[L^2 - (x^2+y^2)\right] \times \lambda & (\kappa_3) \end{cases} \tag{4}$$

where the unknowns (also called *dependent variables*) are the position (x, y) of the ball in Cartesian coordinates and the rope tension λ.

The subsystem $(\kappa_1, \kappa_2, \kappa_3)$ expresses that the distance of the ball from the origin is less than or equal to L, the tension is nonnegative, and one cannot have a distance less than L and a nonzero tension at the same time. This is known as a *complementarity condition*, written as $0 \leq L^2 - (x^2+y^2) \perp \lambda \geq 0$ in the *nonsmooth systems* literature [1], and is an adequate modeling of ideal valves, diodes, and contact in mechanics. Constraints κ_1 and κ_2 are unilateral, which does not

[2] https://www.wolfram.com/mathematica/new-in-9/advanced-hybrid-and-differential-algebraic-equations/

belong to our framework of guarded equations. Therefore, using a technique communicated to us by H. Elmqvist and M. Otter, we redefine the graph of this complementarity condition as a parametric curve, represented by the following three equations:

$$\begin{array}{rcl} s & = & \texttt{if } \gamma \texttt{ then } -\lambda \texttt{ else } L^2-(x^2+y^2) \\ 0 & = & \texttt{if } \gamma \texttt{ then } L^2-(x^2+y^2) \texttt{ else } \lambda \\ \gamma & = & [s \leq 0] \end{array}$$

which allows us to rewrite the complementarity condition $(\kappa_1, \kappa_2, \kappa_3)$ as follows:

$$\left\{ \begin{array}{rll} & 0 = x'' + \lambda x & (e_1) \\ & 0 = y'' + \lambda y + g & (e_2) \\ & \gamma = [s \leq 0] & (k_0) \\ \texttt{if } \gamma \texttt{ then} & 0 = L^2-(x^2+y^2) & (k_1) \\ \texttt{and} & 0 = \lambda + s & (k_2) \\ \texttt{if not } \gamma \texttt{ then} & 0 = \lambda & (k_3) \\ \texttt{and} & 0 = (L^2-(x^2+y^2)) - s & (k_4) \end{array} \right. \tag{5}$$

Unsurprinsingly (considering the reasons for the wrong handling of the clutch example), the Modelica tools also fail to handle this model correctly.

Some new issues emerge from Example (5). First, subsystem $(\kappa_1, \kappa_2, \kappa_3)$ of (4) leaves the impact law at mode change insufficiently specified: it could be fully elastic, fully inelastic, or inbetween. Second, (5) exhibits a logico-numerical fixpoint equation in s, which we regard as problematic. While the Jacobian matrix yields a regularity criterion for systems of smooth algebraic equations, logico-numerical systems of equations are unfriendly: there is no simple criterion for the existence and/or uniqueness of solutions. Note that a sensible modification of model (5) would consist in replacing, in (k_0), s by its left-limit $s^-(t) =_{\text{def}} \limsup_{u \nearrow t} s(u)$.[3]

Our objectives For this example, our objectives are *to address the above difficulties at compile time, and to generate correct simulation code once structural problems are fixed.* These objectives are addressed in Section 3.

1.2.3 A Westinghouse air brake

The second example is a simplified model of the railway air brake patented by George Westinghouse in 1868. A railway car is equipped with the pneumatic system whose schematics is shown in Figure 7. It has two pneumatic ports that are interconnected when several cars are coupled to form a train; in this simple example, only one car is considered, with the boundary conditions shown (`control` is an input pressure). As for the clutch example, this program is not correctly executed by the Modelica 3.3 engine. Using the smoothed version of a valve from the Modelica library of predefined components solves the problem. This, however, requires a careful tuning in accordance with the time scales of the whole system.

A simplified model corresponding to the schematics of Figure 7 is given in Figure 8, where we have taken the boundary conditions into account — the equations are simplified with reference to hydraulics, but the model structure (states and index) are correct. The model of the ideal valve is given by the subsystem (ν_1, ν_2, ν_3). Inequalities (ν_1, ν_2) express nonnegativity conditions satisfied by the pressure difference and the flow through the valve. Equality (ν_3) expresses that, either the valve is open (no pressure drop), or it is closed (no air pass). This is known as a *complementarity condition*, written

$$0 \leq p_r - p_t \;\perp\; f_v \geq 0$$

[3] The left-limit is available in Simulink under the 'state port' construct, whereas the `pre` keyword in Modelica serves the same purpose.

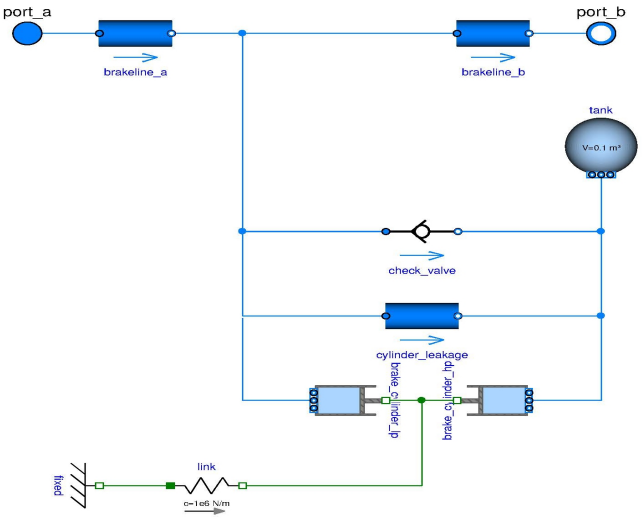


Figure 7: Westinghouse brake for one car: schematics.

p_b : connector pressure (a control)
p_{bn} : next connector pressure
p_r : railcar pressure
p_t : reservoir pressure
f_b : mass flow brake connector
f_{bn} : mass flow next brake connector $f_{bn} = 0$
f_v : mass flow one way valve
f_l : leakage mass flow
f_{cl} : mass flow low press side of cylinder
f_{ch} : mass flow high press side of cylinder
f_t : mass flow reservoir
b : brake force
x : position of the piston

(m_1, m_2) : Mass balance equations
$(f_1\text{–}f_3)$: Flow equations
(r) : Dynamics of the reservoir
(p_1, p_2) : Dynamics of the piston
(l_1, l_2) : Brake force and mech link movement
$(\nu_1\text{–}\nu_3)$: One way ideal valve

$$\begin{cases} 0 = f_b - f_v - f_{cl} + f_l & (m_1) \\ 0 = f_v - f_{ch} - f_l - f_t & (m_2) \\ 0 = f_b - F_1.\varphi(p_b - p_r) & (f_1) \\ 0 = F_1.\varphi(p_{bn} - p_r) & (f_2) \\ 0 = f_l - F_2.\varphi(p_t - p_r) & (f_3) \\ 0 = f_t - \frac{\rho.V}{P_0}.p_t' & (r) \\ 0 = \rho.S.(x'.p_r + (x - L).p_r') + P_0.f_{cl} & (p_1) \\ 0 = \rho.S.(x'.S.p_t + x.p_t') - P_0.f_{ch} & (p_2) \\ 0 = S.(p_t - p_r) - b & (l_1) \\ 0 = K.x - b & (l_2) \\ 0 \le p_r - p_t & (\nu_1) \\ 0 \le f_v & (\nu_2) \\ 0 = f_v \times (p_r - p_t) & (\nu_3) \end{cases}$$

Figure 8: Model corresponding to the schematics of Figure 7; p_b, the control, and f_{bn} determined by the boundary condition f_{bn}=0, are known.

in the *nonsmooth systems* literature [1], and is the adequate modeling of systems such as ideal valve, diode, and contact in mechanics. The two constraints (ν_1, ν_2) are unilateral, which does not belong to our framework of guarded equations. Therefore, using a technique communicated to us by H. Elmqvist and M. Otter, we rewrite the complementarity condition (ν_1, ν_2, ν_3) by using an auxiliary variable s as follows:

$$\begin{cases} \gamma = [s \le 0] & (v_0) \\ \texttt{if } \gamma \texttt{ then } \quad 0 = p_r - p_t & (v_1) \\ \texttt{and} \quad 0 = f_v + s & (v_2) \\ \texttt{if not } \gamma \texttt{ then } \quad 0 = f_v & (v_3) \\ \texttt{and} \quad 0 = p_r - p_t - s & (v_4) \end{cases} \tag{6}$$

The system of Figure 8, where (ν_1, ν_2, ν_3) is replaced by $(v_0, \ldots, v_4)$ in (6), has the form (1). In this model, the subsystem collecting equations $(v_0, \ldots, v_4)$ constitutes a logico-numerical fixpoint equation, in that γ depends on s and the equations determining s are guarded by γ. Given that no

solver exists that reliably supports this kind of fixpoint equation, we think it should be rejected at compile time, with a proper diagnosis. Still, our model is physically meaningful.

Our objectives: For this example, *our objectives are to be able to reject, with a proper diagnosis, the model of Figure 8 (using (6) for the valve), and to correct it so that the corrected model simulates as expected.* These objectives are addressed in Section 4.

1.3 Related work

There exists a huge litterature devoted to the multi-physics DAEs (having a single mode); see, e.g., [16] as a basic reference. Many references related to the Modelica language can also be found on the website of the Modelica organization.[4] We focus this review on works addressing multimode DAE systems with an emphasis on the compilation and code generation, for related models.

A large body of work was developed in the domain of multi-body systems with contacts where impulses can occur; see for instance [44, 43] for an overview of this domain. The current research focuses on the handling of contacts between many bodies using time-stepping methods. Contact equations are usually expressed on the velocity or acceleration, and since the constraints on position level are not explicitly taken into account, a drift typically occurs. An exception is [50], where constraints on position level are enforced. It is not obvious how the specialized multimode methods for multi-body systems can be generalized to any type of multimode DAE. For electrical circuits with idealized switches, like ideal diodes, impulses can also occur. Again, a large literature is available, concentrating mostly on specialized electrical circuits, such as piecewise-linear networks with idealized switches [30]. However, it is not obvious how to generalize methods in this restricted area to general multimode DAEs.

The article by Mehrmann *et al.* [38] contains interesting results regarding numerical techniques to detect chattering between modes. It however assumes that consistent reset values are explicitly given for each mode. Such an assumption does not hold in general, particularly for models derived from first principles of the physics; the clutch and cup-and-ball examples are good illustrations of this.

In the PhD thesis of Zimmer [54], variable-structure multi-domain DAEs are defined thanks to an *ad hoc* modeling language, and a runtime interpreter is used that processes the equations at runtime, when the structure and/or the index changes. Limitations of this work are that impulsive behavior is not supported and that the user has to explicity define the transfer of variable values from one mode to another, which is not practical for large models.

Describing variable-structure systems with *causal* state machines is discussed by Pepper *et al.* [41]. Dynamically changing the structural analysis at runtime is also performed by Hoeger [31, 32]. In [31], the author proposes a dynamic execution of John Pryce's Σ-method [48].

Elmqvist *et al.* [28, 36] propose a high-level description of multimode models as an extension to the synchronous Modelica 3.3 state machines, by using continuous-time state machines having continuous-time models as 'states'. Besides ODEs as used in hybrid automata, *acausal DAE models with physical connectors* can also be a 'state' of a state machine. Such a state machine is mapped into a form such that the resulting equations can be processed by standard symbolic algorithms supported by Modelica tools. The major restrictions of this approach are that mode changes with impulsive behavior are not supported and that not all types of multimode systems can be handled due to the static code generation.

Correct restart at mode changes is clearly a central issue in multi-mode DAE systems. Techniques from *nonsmooth dynamical systems* [1] address this issue by performing a warm restart at mode changes using time-stepping discretization schemes: not only the numerical scheme does not interrupt at mode changes, but these mode changes are not even detected. This is very elegant and effective. However, this relies on the handling of *complementarity conditions* (see Section 1.2.2), thus, it does not apply to general multimode DAE systems but only to a certain subclass.

[4] https://www.modelica.org/publications

We observe that, in general, the issue of correct restart at mode changes is not addressed in the above mentioned literature. It is, however, considered in the few references to follow.

Benveniste *et al.* [7, 6] considered this issue, as well as the problem of varying structure and index, from a fundamental point of view, by relying on nonstandard analysis to capture continuous-time dynamics and mode change events in a unified framework. A first structural analysis algorithm was presented in [7], by significantly modifying the original Pantelides algorithm [40]. A proof-of-concept mockup named SunDAE was developed implementing this approach. This first attempt suffers from some issues: the proposed structural analysis does not boil down trivially to the Pantelides algorithm in the case of single-mode systems; it involves nondeterministic decisions, an unwanted feature for the mathematical foundation of compilers; and its mathematical study is incomplete.

The work of Stephan Trenn is an important contribution to the subject. In his PhD thesis [52] and his article [53], he points out the difficulty in defining piecewise smooth distributions: there is no intrinsic definition for their 'Dirac' part, as several definitions comply with the requirements for it. This indicates that distributions are not the ultimate answer to deal with impulsive variables in multimode DAE systems. Still, Trenn was able in [34] 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. A main difference with our approach is that Trenn's solution is complete for all variables and relies on piecewise smooth distributions, whereas our works on restart conditions consist in eliminating impulsive variables while computing restart values for state variables. (Beyond the existence of solutions, the work of Trenn *et al.* contains additional interesting results that are not relevant to our study.)

An important step forward was done in [8]. The interesting subclass of multimode DAE systems that is now called *semi-linear* (see Section 11.2) was first identified. Semi-linear multimode DAE systems can exhibit impulsive variables at mode changes. They generalize the 'semi-linear systems' proposed by Trenn in the sense that switching conditions are no longer restricted to time-based ones, instead including state-based switching conditions. The analysis and discretization schemes proposed in [8] are mathematically sound. Building on this work, Martin Otter has developed the ModiaMath tool for semi-linear multimode DAE systems. Hence, it makes sense to compare the schemes proposed in [8] to the ones we develop in this paper, for general mDAE systems. It turns out that our general approach coincides with the schemes proposed in [8] when applied to the subclass of semi-linear systems, see Section 11.2. Our present work thus extends and significantly improves [8].

1.4 Contributions

In this work, we develop a systematic approach that addresses, as particular cases, the objectives stated for the examples of Sections 1.2.1 and 1.2.2. Its main contributions can be detailed as follows.

Structural analysis of mode changes using nonstandard analysis As our main contribution, we extend the notion of structural analysis to mode change events. More precisely, we develop a structural analysis for multimode DAE systems that is valid at any time, that is, for both continuous dynamics and mode changes.

Doing so raises a number of difficulties. First, mode changes occur under various kinds of dynamics; there can be isolated events or finite cascades thereof, but Zeno situations can also occur, i.e., events of mode changes can accumulate and form an infinite sequence bounded by some finite instant. As a result, sliding modes can emerge out of mode changes, and the modeling and simulation tools can only identify such situations at runtime. Second, impulses may occur at mode changes for certain variables, which again is most of the time discovered by tools at runtime. Compilation, however, is a task performed prior to generating simulation code; hence, it encounters serious difficulties in both the nature of time at mode changes and the domain of variables, due to impulsive behaviors.

The main objective of structural analysis, akin to compilation in general, is precisely to keep

away from numerical considerations, by focusing only on the structure (or the syntax) of the system of equations. This is captured by the bipartite graph associated to the considered DAE system (also referred to as 'incidence graph' or 'incidence matrix' or 'Σ-matrix' depending on the context and authors). The basic principle supporting structural analysis consists in: (1) abstracting an equation $f(x, y, z) = 0$ as the bipartite graph having f, x, y, z as vertices and $(f, x), (f, y), (f, z)$ as edges, and a system of equations as the union of the above bipartite graphs; and (2) exploiting the resulting graph to generate efficient simulation code. This means focusing on syntax and symbols, while abstracting away numerical aspects. We thus wish to *symbolically* manipulate both normal and impulsive values for variables, as well as both continuous time and events (in finite or infinite cascades), and possibly stranger structures for time.

We see no vehicle for supporting all of this, other than *nonstandard analysis*. Nonstandard analysis was proposed by Abraham Robinson in the 1960s to allow the explicit manipulation of 'infinitesimals' and 'infinities' in analysis [49, 23]. Impulsive values become normal citizens in nonstandard analysis. Continuous time and dynamics can be discretized using infinitesimal time steps, e.g., by setting $x'(t) \approx \frac{x(t+\partial)-x(t)}{\partial}$ with ∂ infinitesimal, thus providing perfect fidelity up to infinitesimal errors. This discretized time can then support the various kinds of time needed when modeling mode changes. Since techniques from structural analysis translate almost verbatim to discrete-time dynamics, having dynamical systems indexed by discrete (nonstandard) time in a uniform setting paves the way toward structural analysis for multimode DAE systems in their generality. Using this approach, we are able to extend the structural analysis, from single-mode to multimode DAE systems, by encompassing both modes and mode changes.

Generating code for the restart at mode changes The structural analysis developed hereafter works in the nonstandard analysis domain. As such, it cannot be executed on any real-life computer. A further stage is needed to produce executable code, by mapping the nonstandard execution schemes to ordinary world (standard) schemes; this is called *standardization*. This allows us to produce mathematically justified schemes for the restart at mode changes.

Rejecting and accepting programs on a clear basis Our structural analysis is precise enough to properly identify models that are structurally over- or under-specified at mode change events, or models that contain a fixpoint equation involving a subset of variables and some guard. We reject such models, with a proper diagnosis explaining the rejection. In turn, mode-dependent index/state/dynamics are not reasons for rejection *per se*. Our compilation technique is able to handle such cases.

The paper is organized as follows This report is an extended version of a paper under submission. With reference to this submission, it contains additional material, which the reader may skip for a first reading. In the following list of sections, we indicate in blue these additional sections.

The clutch example is informally developed in Section 2; we show how to map this model to the nonstandard domain, how the structural analysis works (in particular, the handling of mode changes is emphasized), and how standardization is performed to generate actual simulation code. The Cup-and-Ball example is developed in Section 3, following the same guidelines; based on the case of elastic impact, it brings forth the need for considering transient modes, i.e., modes that last for zero time. Section 4 develops the Westinghouse air brake example; we address the objectives stated in the introduction and we advocate for the interest of adding assertions, restricting the allowed sequences of modes in the system. The insight gained from studying these examples leads to the approach described in Section 5.

The extension of structural analysis to multimode DAE systems is then developed. In Section 6, the basics of structural analysis for algebraic systems of equations are recalled. Then, we review J. Pryce's Σ-method for the structural analysis of DAE systems. We extend the Σ-method to the multimode case, including the handling of mode change events, in Section 7, for the subclass of systems involving only long modes—which excludes the Cup-and-Ball example with elastic impact.

Motivated by the latter case, we identify the need for handling *transient* modes, in which the system spends zero time; this is addressed in Section 7.4. In Section 8 we widen our scope by adding to our framework, in addition to derivatives, the left- and right-limits of a signal, which is a useful primitive operator provided by modeling languages such as Modelica or Simulink.

Section 9 recalls the needed background on nonstandard analysis, mostly related to differential calculus. In Section 10, we complement this background with additional results related to structural analysis and impulse analysis, and we apply this machinery to the task of standardization.

Three important results are developed in Section 11.

- In Section 11.1, we prove that modifying the nonstandard expansion of the derivative (there are infinitely many possibilities) does not change the final generated code, thus showing that our approach is intrinsic.
- We prove in Section 11.2 that our approach does compute correct solutions for *semi-linear* multimode DAE systems, a nontrivial subclass possibly involving impulsive behaviors and containing, as a subclass, multi-body mechanics (see [8]). We would be happy to prove that the simulation code we generate does compute the solution of any multimode DAE system; this, however, is beyond what is doable since no notion of solution is known for multimode DAE systems in general.
- We propose in Section 11.3 a numerical scheme for approximating restart values for non-impulsive variables at mode changes. This alleviates the need for performing standardization when generating code in practice; the proof of this result, however, uses standardization as a mathematical tool.

Finally, we draw in Section 12 how our approach can be implemented in a tool, and we illustrate this on the RLDC2 circuit example. This example, provided to us by Sven-Erik Mattsson, is problematic for the existing Modelica tools, as it has mode-dependent index and structure. The structural analysis of the corresponding DAE system in all its modes is successfully handled by our IsamDAE tool [19]. This tool, currently under development, will support multimode DAE structural analysis following our theory. Its current version does not support impulsive variables at mode changes—the RLDC2 circuit example, however, does not exhibit this difficulty.

In Section 7, we assumed that the effect of guards is delayed by one nonstandard time shift. This is a necessary condition for being able to support both long and transient modes. In Apprendix A, we propose an alternative execution scheme that is *constructive* in that it can handle models in which the effect of guards is not delayed by one nonstandard time shift. In turn, only models with long modes are supported.

2 The ideal clutch

In this section, we introduce our approach by discussing in detail the ideal clutch example presented in Section 1.2.1. Its model was given in (3). We first analyze separately the model for each mode of the clutch (Section 2.1). Then, we discuss the difficulties arising when handling mode changes (Section 2.2). Finally, we propose a global comprehensive analysis in Sections 2.3 – 2.5. For convenience, we present an intuitive and informal introduction to nonstandard analysis in Section 2.3.

2.1 Separate Analysis of Each Mode

In the released mode, when γ is false in System (3), the two shafts are independent and one obtains the following two independent ODEs for ω_1 and ω_2:

$$\begin{array}{llll} \omega_1' = f_1(\omega_1, \tau_1) & (e_1) & \tau_1 = 0 & (e_5) \\ \omega_2' = f_2(\omega_2, \tau_2) & (e_2) & \tau_2 = 0 & (e_6) \end{array} \tag{7}$$

In the engaged mode, however, γ holds true, and both the two velocities and the two torques are algebraically related:

$$\begin{array}{llll} \omega_1' = f_1(\omega_1, \tau_1) & (e_1) & \omega_1 - \omega_2 = 0 & (e_3) \\ \omega_2' = f_2(\omega_2, \tau_2) & (e_2) & \tau_1 + \tau_2 = 0 & (e_4) \end{array} \tag{8}$$

Equation (e_4), that relates the two torques, is not an issue by itself, as it can be rewritten as $\tau_2 = -\tau_1$ and used to eliminate τ_2. Equation (e_3), however, relates the two velocities ω_1 and ω_2 that are otherwise subject to the ODE system (e_1, e_2), which makes System (8) a DAE instead of an ODE.

In System (8), the torques τ_1 and τ_2 are not differentiated: we call them *algebraic variables*. The derivatives of the velocities ω_1 and ω_2 appear: they are *state variables*. The $\omega_1', \omega_2', \tau_1, \tau_2$ are called the *leading variables* of System (8).

Now, suppose one is able to uniquely determine the leading variables $\omega_1', \omega_2', \tau_1, \tau_2$ 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" [48].

It turns out that this condition does not hold for System (8) as is. To intuitively explain the issue, we move to discrete time by applying an explicit first-order Euler scheme with constant step size $\delta > 0$:

$$\begin{array}{llll} \omega_1^\bullet = \omega_1 + \delta \cdot f_1(\omega_1, \tau_1) & (e_1^\delta) & \omega_1 - \omega_2 = 0 & (e_3) \\ \omega_2^\bullet = \omega_2 + \delta \cdot f_2(\omega_2, \tau_2) & (e_2^\delta) & \tau_1 + \tau_2 = 0 & (e_4) \end{array} \tag{9}$$

where $\omega^\bullet(t) =_{\text{def}} \omega(t+\delta)$ denotes the forward time shift operator by an amount of δ. Suppose we are given consistent values $\omega_1 = \omega_2$ and we wish to use System (9), 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 dependent variable.

However, since System (9) is time invariant, and assuming that the system remains in the engaged mode for at least δ seconds, there exists an additional *latent equation* on the set of variables, namely

$$\omega_1^\bullet - \omega_2^\bullet = 0 \qquad (e_3^\bullet) \tag{10}$$

obtained by shifting (e_3) forward. Now, replacing (e_3) with $(e_3^\bullet)$ in System (9) yields a system with four equations and four dependent variables; moreover, this system is, in fact, nonsingular in a generic sense. One can now use System (9) along with equation $(e_3^\bullet)$ to get an execution scheme for the engaged mode of the clutch. This is shown in Exec. Sch. 1 below.

Execution Scheme 1 System (9)+Eq. (10).

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 new values of the state variables satisfy (10) by construction, the consistency condition is met at the next iteration step (should the system remain in the same mode).

Comment 1 (structural nonsingularity) The implicit assumption behind Line 1 in Exec. Sch. 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 the parameters in the considered equations (through the unspecified functions f_1 and f_2). We will repeatedly use *structural nonsingularity* (also called *structural regularity*) in the sequel. A formalization of structural reasoning and analysis is developed in Section 6. □

Observe that the same analysis could be applied to the original continuous-time dynamics from System (8) by augmenting it with the following *latent equation*:

$$\omega_1' - \omega_2' = 0 \qquad (e_3') \tag{11}$$

obtained by differentiating (e_3); as a matter of fact, 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 Exec. Sch. 2, that parallels Exec. Sch. 1.

Execution Scheme 2 System (8)+Eq. (11).

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 the ω_1 and ω_2 is easy in Exec. Sch. 1, the situation changes in Exec. Sch. 2: the derivatives (ω_1', ω_2') are first evaluated, and then an ODE solver (here denoted by `ODESolve`) is used to update the state (ω_1, ω_2) for the next discretization step. Note that, when considering an exact mathematical solution, if $\omega_1 - \omega_2 = 0$ initially holds and $\omega_1' - \omega_2' = 0$, then the linear constraint (e_3) will be satisfied at any positive time.

The replacement of (e_3) by (e_3'), which resulted in Exec. Sch. 2, is known in the literature as *index reduction* [37]. It requires adding the (smallest set of) latent equations needed for the system in Line 1 of the execution scheme to become solvable and deterministic. The number of successive time differentiations needed for obtaining all the latent equations is called the *differentiation index* [21], which we simply refer to as the *index* in the sequel.[5] Finally, observe that both execution schemes rely on an algebraic equation system solver to evaluate the leading variables as a function of state variables.

We conclude this part by briefly discussing the initialization problem. Unlike for ODE systems, initialization is far from trivial for DAE systems. For the clutch example, if one considers System (8) as a standalone DAE, the initialization is performed as indicated in Exec. Sch. 3.

Execution Scheme 3 Initialization of System (8)+Eq. (10).

1: $(\omega_1, \omega_2) \leftarrow$ `Solve`$\{e_3\}$ for (ω_1, ω_2) ▷ 2 unknowns for only 1 equation

The system for solving possesses two unknowns for only one equation, which leaves us with one degree of freedom; in mathematical terms, the set of all initial values for the 2-tuple of variables is a manifold of dimension 1, generically or structurally. For example, one may freely fix the initial common rotation speed so that (e_3) is satisfied.

2.2 Mode Transitions

In an attempt to reduce the full clutch model to the analysis of the DAE of each mode, one hopes that the handling of a mode change boils down to applying the initialization given in Exec. Sch. 3. If one was to handle restarts at mode changes like initializations, it would make the clutch system nondeterministic, due to the *extra* degree of freedom of Exec. Sch. 3. In contrast, the physics tells us that the state of the system is entirely determined when the clutch gets engaged. This comforts the intuition that resets at mode changes strongly differ from mere initializations.

However, inferring by hand the reset values for rotation velocities when the clutch gets engaged is definitely non-trivial. Furthermore, these values depend on the whole system model, so that the

[5] There are indeed a great number of possible definitions for "the index" of a DAE, serving various purposes, see [21] for instance. This, however, is not relevant to the present work.

task of determining them becomes complex if external components are added. *It is therefore highly desirable, for this example, to let the compiler infer these reset values from model* (3).

This problem involves a mix of continuous-time and discrete-time dynamics, the latter consisting in one or several computation steps to restart the dynamics of the next mode. As a result, it will be convenient to cast everything in discrete time thanks to nonstandard analysis. We develop this next.

2.3 Nonstandard Semantics

Nonstandard analysis [49, 35, 5] extends the set $\mathbb{R}$ of real numbers into a superset ${}^\star\mathbb{R}$ of *hyperreals* (also called *nonstandard reals*) that includes infinite sets of infinitely large numbers and infinitely small numbers. The key properties of hyperreals, needed for the informal discussion of the clutch example, are the following [35]:[6]

- There exist *infinitesimals*, defined as hyperreals that are smaller in absolute value than any real number: an infinitesimal $\partial \in {}^\star\mathbb{R}$ is such that $|\partial| < a$ for any positive $a \in \mathbb{R}$. For x, y two hyperreals, write $x \approx y$ if $x - y$ is an infinitesimal.
- All relations, operators, and propositional formulas that are valid over $\mathbb{R}$ are also valid over ${}^\star\mathbb{R}$; for example, ${}^\star\mathbb{R}$ is a totally ordered set. The arithmetic operations $+$, $\times$, etc. can be lifted to ${}^\star\mathbb{R}$. We say that a hyperreal x is *finite* if there exists some standard finite positive real number a such that $|x| < a$.
- For every finite hyperreal $x \in {}^\star\mathbb{R}$, there exists a unique standard real number $\mathsf{st}(x) \in \mathbb{R}$ such that $\mathsf{st}(x) \approx x$, and $\mathsf{st}(x)$ is called the *standard part* (or *standardization*) of x. As we shall see, standardizing more complex objects, such as functions or systems of equations, requires some care.
- Every real function lifts in a systematic way to a hyperreal function (regardless of its continuity properties).[7] This allows us to write $f(x)$ where f is a real function and x is a nonstandard number.
- Let $t \mapsto x(t), t \in \mathbb{R}$ be an $\mathbb{R}$-valued (standard) signal. Then:

$$x \text{ is continuous at instant } t \in \mathbb{R} \text{ if and only if, for any infinitesimal } \partial \in {}^\star\mathbb{R}, \text{ one has } x(t+\partial) \approx x(t); \tag{12}$$

$$x \text{ is differentiable at instant } t \in \mathbb{R} \text{ if and only if there exists } a \in \mathbb{R} \text{ such that, for any infinitesimal } \partial \in {}^\star\mathbb{R}, \frac{x(t+\partial)-x(t)}{\partial} \approx a. \text{ In this case, } a = x'(t). \tag{13}$$

We can then consider, for a given positive infinitesimal ∂, the following time index set $\mathbb{T} \subseteq {}^\star\mathbb{R}$:

$$\mathbb{T} = 0, \partial, 2\partial, 3\partial, \cdots = \{n\partial \mid n \in {}^\star\mathbb{N}\} \tag{14}$$

where ${}^\star\mathbb{N}$ denotes the set of *hyperintegers*, consisting of all natural integers augmented with additional infinite numbers called *nonstandard*. For convenience, the elements of $\mathbb{T}$ will be generically denoted by the symbol t, the explicit expression $n\partial$, or even simply by t when the possible confusion with (usual) real time does not matter. The important features of $\mathbb{T}$ are:[8]

- Any finite positive real time $t \in \mathbb{R}, t \geq 0$ is infinitesimally close to some element of $\mathbb{T}$ (informally, $\mathbb{T}$ covers the set $\mathbb{R}_+$ of nonnegative real numbers and can be used to index continuous-time dynamics); and

[6] The skeptical reader is gently referred to Section 9 for mathematical details.
[7] The result of this lifting may not be intuitive in general, see Section 9 for mathematical details.
[8] In French, one uses to qualify this as: *avoir le beurre et l'argent du beurre.*

- $\mathbb{T}$ is 'discrete': every instant $n\partial$ has a predecessor $(n-1)\partial$ and a successor $(n+1)\partial$, except 0 that has no predecessor.

Definition 2 (Forward and Backward Shifts) *Let x be a nonstandard signal indexed by $\mathbb{T}$. We define the forward shifted signal $x^\bullet$ through $x^\bullet(n\partial) =_{\text{def}} x((n+1)\partial)$ and, for $f(X)$ a function of the tuple X of signals, we set $(f(X))^\bullet =_{\text{def}} f(X^\bullet)$ where the forward shift $X \mapsto X^\bullet$ applies pointwise to all the components of the tuple. It will also be convenient to consider the* backward shift $^\bullet x$ *defined by $^\bullet x((n+1)\partial) =_{\text{def}} x(n\partial)$, implying that an initial value for $^\bullet x(0)$ must be provided.*

For example, $f^\bullet(x,y)(t) =_{\text{def}} f(x^\bullet, y^\bullet)(t) = f(x(t+\partial), y(t+\partial))$. Observe that the definition of the forward shift depends on the infinitesimal ∂. Thanks to (13), using forward shifts allows us to represent, up to an infinitesimal, the derivative x' of a signal by its first-order explicit Euler approximation $\frac{1}{\partial}(x^\bullet - x)$.

Solutions of multimode DAE systems may, however, be non-differentiable and even non-continuous at events of mode change. To give a meaning to x' at any instant, *we decide to **define** it everywhere as the nonstandard first-order Euler increment*

$$x' =_{\text{def}} \frac{1}{\partial}(x^\bullet - x) \ . \tag{15}$$

Definition 3 *Substituting, in a multi-mode DAE system, every occurrence of $x'(t)$ by its expansion (15) yields a difference algebraic equation* (dAE) *system*[9] *that we call its* nonstandard semantics.

For instance, the nonstandard semantics of System (3) is the following:

$$\left\{\begin{array}{rll} & \frac{\omega_1^\bullet - \omega_1}{\partial} = f_1(\omega_1, \tau_1) & (e_1^\partial) \\ & \frac{\omega_2^\bullet - \omega_2}{\partial} = f_2(\omega_2, \tau_2) & (e_2^\partial) \\ \texttt{if } \gamma \ \texttt{ then} & \omega_1 - \omega_2 = 0 & (e_3) \\ \texttt{and} & \tau_1 + \tau_2 = 0 & (e_4) \\ \texttt{if not } \gamma \ \texttt{ then} & \tau_1 = 0 & (e_5) \\ \texttt{and} & \tau_2 = 0 & (e_6) \end{array}\right. \tag{16}$$

The state variables are ω_1, ω_2 whereas the leading variables are now τ_1, τ_2, $\omega_1^\bullet$, $\omega_2^\bullet$, in both modes $\gamma = \text{F}$ and $\gamma = \text{T}$. Reproducing the reasoning of Section 2.1, one obtains, for each mode, a discrete system very much like the explicit Euler scheme of Section 2.1, except that the step size is now infinitesimal and that the variables are all nonstandard. The added value of System (16) with respect to the explicit Euler scheme of Section 2.1 is twofold: first, it is exact up to infinitesimals within each mode; second, it will allow us to carefully analyze what happens at events of modes change.

2.4 Nonstandard structural analysis

We now develop a structural analysis for the nonstandard multimode dAE system (16). As a preliminary step, we perform the structural analysis in each mode and combine the two dynamics with the due guards (the added latent equation is shown in red):

$$\left\{\begin{array}{rll} & \frac{\omega_1^\bullet - \omega_1}{\partial} = f_1(\omega_1, \tau_1) & (e_1^\partial) \\ & \frac{\omega_2^\bullet - \omega_2}{\partial} = f_2(\omega_2, \tau_2) & (e_2^\partial) \\ \texttt{if } \gamma \ \texttt{ then} & \omega_1 - \omega_2 = 0 & (e_3) \\ \texttt{and} & \color{red}{\omega_1^\bullet - \omega_2^\bullet = 0} & \color{red}{(e_3^\bullet)} \\ \texttt{and} & \tau_1 + \tau_2 = 0 & (e_4) \\ \texttt{if not } \gamma \ \texttt{ then} & \tau_1 = 0 & (e_5) \\ \texttt{and} & \tau_2 = 0 & (e_6) \end{array}\right. \tag{17}$$

System (17) coincides with the known structural analysis in the interior of each of the two modes $\gamma = \text{T}$ and $\gamma = \text{F}$. What about the instants of mode change?

[9] Throughout this paper, we consistently use, in acronyms, the letters "D" and "d" to refer to "Differential" and "difference", respectively.

Mode change $\gamma : \mathrm{T} \to \mathrm{F}$ At the considered instant, we have ${}^\bullet\gamma = \mathrm{T}$ and $\gamma = \mathrm{F}$, where ${}^\bullet\gamma$ denotes the backward shift of γ following Definition 2. Unfolding System (17) at the two successive previous (shown in blue) and current (shown in black) instants yields, by taking the actual values for the guard at those instants into account:

$$\begin{array}{ll} \text{previous} & \left\{\begin{array}{ll} \frac{\omega_1 - {}^\bullet\omega_1}{\partial} = f_1({}^\bullet\omega_1, {}^\bullet\tau_1) & \\ \frac{\omega_2 - {}^\bullet\omega_2}{\partial} = f_2({}^\bullet\omega_2, {}^\bullet\tau_2) & \\ {}^\bullet\omega_1 - {}^\bullet\omega_2 = 0 & ({}^\bullet e_3) \\ \omega_1 - \omega_2 = 0 & \\ {}^\bullet\tau_1 + {}^\bullet\tau_2 = 0 & \end{array}\right. \\ \text{current} & \left\{\begin{array}{l} \frac{\omega_1^\bullet - \omega_1}{\partial} = f_1(\omega_1, \tau_1) \\ \frac{\omega_2^\bullet - \omega_2}{\partial} = f_2(\omega_2, \tau_2) \\ \tau_1 = 0 \\ \tau_2 = 0 \end{array}\right. \end{array} \tag{18}$$

We assume that values are given for the state variables at the previous instant, namely ${}^\bullet\omega_1$ and ${}^\bullet\omega_2$, and we regard System (18) as an algebraic system of equations with dependent variables ${}^\bullet\tau_i, \omega_i; \tau_i, \omega_i^\bullet$ for $i = 1, 2$, i.e., the leading variables of System (17) at the previous and current instants. Note that equation $({}^\bullet e_3)$ does not involve any dependent variable: it is actually a *fact* that is inherited from the instant $t - 2\partial$, where t is the current instant; we can thus ignore this equation in System (18). Seen this way, System (18) is structurally nonsingular and we can solve it as two successive blocks, corresponding to the previous instant followed by the current instant: the latter is our desired code for the mode change. This yields the following code for the mode change $\gamma : \mathrm{T} \to \mathrm{F}$:

$$\left\{\begin{array}{l} \omega_1, \omega_2 \text{ set by previous instant} \\ \frac{\omega_1^\bullet - \omega_1}{\partial} = f_1(\omega_1, \tau_1) \\ \frac{\omega_2^\bullet - \omega_2}{\partial} = f_2(\omega_2, \tau_2) \\ \tau_1 = 0 \\ \tau_2 = 0 \end{array}\right. \tag{19}$$

The above reasoning may seem an overshoot for this mode change $\gamma : \mathrm{T} \to \mathrm{F}$, since one could simply say: let the previous part of System (18) fix ω_1, ω_2 and, then, let the current part of System (18) determine $\tau_i, \omega_i^\bullet$ for $i = 1, 2$. However, the same line of reasoning will allow us to address the other, more difficult, mode change as well.

Mode change $\gamma : \mathrm{F} \to \mathrm{T}$ At the considered instant, we have ${}^\bullet\gamma = \mathrm{F}$ and $\gamma = \mathrm{T}$. We proceed as for the other case. Unfolding System (17) at the two successive previous (shown in blue) and current (shown in black) instants yields, by taking the actual values for the guard at those instants into account:

$$\begin{array}{ll} \text{previous} & \left\{\begin{array}{ll} \frac{\omega_1 - {}^\bullet\omega_1}{\partial} = f_1({}^\bullet\omega_1, {}^\bullet\tau_1) & ({}^\bullet e_1^\partial) \\ \frac{\omega_2 - {}^\bullet\omega_2}{\partial} = f_2({}^\bullet\omega_2, {}^\bullet\tau_2) & ({}^\bullet e_2^\partial) \\ {}^\bullet\tau_1 = 0 & \\ {}^\bullet\tau_2 = 0 & \end{array}\right. \\ \text{current} & \left\{\begin{array}{ll} \frac{\omega_1^\bullet - \omega_1}{\partial} = f_1(\omega_1, \tau_1) & \\ \frac{\omega_2^\bullet - \omega_2}{\partial} = f_2(\omega_2, \tau_2) & \\ \omega_1 - \omega_2 = 0 & (e_3) \\ \omega_1^\bullet - \omega_2^\bullet = 0 & \\ \tau_1 + \tau_2 = 0 & \end{array}\right. \end{array} \tag{20}$$

Once more, we regard System (20) as an algebraic system of equations with dependent variables ${}^\bullet\tau_i, \omega_i; \tau_i, \omega_i^\bullet$ for $i = 1, 2$, i.e., the leading variables of System (17) at the previous and current instants.

In contrast to the previous mode change, System (20) is no longer structurally nonsingular since the following subsystem possesses five equations and only four dependent variables $\omega_1, \omega_2, {}^\bullet\tau_1, {}^\bullet\tau_2$:

$$\begin{cases} \frac{\omega_1 - {}^\bullet\omega_1}{\partial} = f_1({}^\bullet\omega_1, {}^\bullet\tau_1) & ({}^\bullet e_1^\partial) \\ \frac{\omega_2 - {}^\bullet\omega_2}{\partial} = f_2({}^\bullet\omega_2, {}^\bullet\tau_2) & ({}^\bullet e_2^\partial) \\ {}^\bullet\tau_1 = 0 & \\ {}^\bullet\tau_2 = 0 & \\ \omega_1 - \omega_2 = 0 & (e_3) \end{cases} \tag{21}$$

This subsystem exhibits a conflict. We propose to resolve it by applying the following causality principle:

Principle 1 (causality) *What was done at the previous instant cannot be undone at the current instant.*

Applying Principle 1 leads to removing, from subsystem (21), the conflicting equation (e_3). This yields the following code for the mode change $\gamma : \text{F} \to \text{T}$:

$$\begin{cases} \omega_1, \omega_2, {}^\bullet\tau_1, {}^\bullet\tau_2 \text{ set by previous instant} \\ \frac{\omega_1^\bullet - \omega_1}{\partial} = f_1(\omega_1, \tau_1) \\ \frac{\omega_2^\bullet - \omega_2}{\partial} = f_2(\omega_2, \tau_2) \\ \omega_1^\bullet - \omega_2^\bullet = 0 \\ \tau_1 + \tau_2 = 0 \end{cases} \tag{22}$$

Note that the consistency equation $(e_3) : \omega_1 - \omega_2 = 0$ has been removed from System (22), thus modifying the original model. However, this removal occurs only at mode change events $\gamma : \text{F} \to \text{T}$. What we have done amounts to *delaying by one nonstandard instant the satisfaction of some of the constraints in force in the new mode* $\gamma = \text{T}$. Since our time step ∂ is infinitesimal, this takes zero standard time.

The corresponding nonstandard execution scheme is summarized in Exec. Sch. 4. We use variable Δ to encode the *context*, that is, the set of equations known to be satisfied by the state variables as a result of having executed the previous instant. For instance, $e_3 \notin \Delta$ corresponds to the mode change $\gamma : \text{F} \to \text{T}$, expressing that (e_3) is not provably true if the guard was false at the previous instant. At each tick, that is, every time the scheme moves to a new instant, the context gets updated to account for the equations satisfied by the new state. The procedure '`Solve`' solves the (algebraic) system to determine the values of the leading variables.

Execution Scheme 4 for Nonstandard System (16). Comments are written in blue.

Require: ω_1 and ω_2.
1: **if** γ (engaged mode) **then**
2: **if** $e_3 \notin \Delta$ (mode change) **then**
3: $(\omega_1^\bullet, \omega_2^\bullet) \leftarrow$ `Solve` $\{e_1^\partial, e_2^\partial, e_3^\bullet, e_4\}$
4: `Tick`: $\Delta \leftarrow \Delta \cup \{e_3\}$
5: **else**
6: $(\tau_1, \tau_2, \omega_1^\bullet, \omega_2^\bullet) \leftarrow$ `Solve` $\{e_1^\partial, e_2^\partial, e_3^\bullet, e_4\}$
7: `Tick`: Δ unchanged
8: **else** (released mode)
9: $(\tau_1, \tau_2, \omega_1^\bullet, \omega_2^\bullet) \leftarrow$ `Solve` $\{e_1^\partial, e_2^\partial, e_5, e_6\}$
10: `Tick`: $\Delta \leftarrow \Delta \setminus \{e_3\}$

Observe that Exec. Sch. 4 would work without changes if the guard γ was a predicate on the state variables ω_1 and ω_2.

2.5 Standardization

Exec. Sch. 4 cannot be run in its present form, since it involves nonstandard reals. To recover executable code over the real numbers, a supplementary *standardization* step is needed. Recall that any finite nonstandard real x has a unique standard part $\mathsf{st}(x) \in \mathbb{R}$ such that $x \approx \mathsf{st}(x)$. The standardization procedure aims at recovering the standard parts of the leading variables from their nonstandard version. We distinguish two cases: continuous evolutions within each mode, assuming that the sojourn time in each mode has (standard) positive duration, and discrete evolutions at events of mode change.

2.5.1 Within continuous modes

Let $x : t \mapsto x(t)$, $t \in [s, p)$, denote the real continuous and differentiable solution in a given mode (assuming it exists). In the sequel, we use the notation $o(\xi)$ to denote any (unspecified) nonstandard real number ζ such that ζ/ξ is infinitesimal. Using (13) and this notation, equations (e_1^∂) and (e_2^∂) rewrite as $\omega_i' = f_i(\omega_i, \tau_i) + o(1)$, which can be shown to standardize as the differential equations (e_1) and (e_2), respectively. This suffices to recover the dynamics (7) in the released mode $\gamma = \text{F}$.

In the engaged mode $\gamma = \text{T}$, however, we need to handle $(e_3^\bullet) : \omega_1^\bullet - \omega_2^\bullet = 0$. The nonstandard characterization of derivatives writes $\omega_i^\bullet = \omega_i + \partial.\omega_i' + o(\partial)$, where ω_i and ω_i' are both standard. Since $\omega_1 - \omega_2 = 0$ and $\omega_1^\bullet - \omega_2^\bullet = 0$ both hold, we inherit $\omega_1' - \omega_2' = o(1)$, which implies $\omega_1' - \omega_2' = 0$ since the ω_i' are standard. We thus recover the dynamics (8) augmented with the latent equation (11) in the engaged mode $\gamma = \text{T}$.

To summarize, within continuous modes, we perform structural analysis as usual for DAE systems in continuous time: nothing new happens.

2.5.2 At the instants of mode change

Suppose we have an event of mode change at time t, meaning that $\gamma(t) \neq \gamma(t - \partial)$. Our aim is to use the next values $\omega_i^\bullet(t) = \omega_i(t + \partial)$ to initialize the state variables for the next mode with the value $\omega_i^+(t) = \omega_i^\bullet(t)$. Thus, unlike for continuous time dynamics, the occurrence of the infinitesimal time step ∂ in the expression $\omega_i(t + \partial)$ is not an issue: it just points to the supplementary instant at which restart is performed.

However, the equations defining $\omega_i^\bullet$ as functions of the ω_i's involve the hyperreal ∂ *in space*: standardization must be performed to get rid of it.

Transition $\gamma : \text{T} \to \text{F}$ It is easy, as the new mode has an ODE dynamics whose state variables are initialized with corresponding exit values when leaving the previous mode.

Transition $\gamma : \text{F} \to \text{T}$ This one is more involved. As established in Exec. Sch. 4, in order to compute the reset values, we use the system of 4 equations $\{e_1^\partial, e_2^\partial, e_3^\bullet, e_4\}$ to determine the leading variables $(\tau_1, \tau_2, \omega_1^\bullet, \omega_2^\bullet)$. In particular, from e_i^∂, we get

$$\frac{\omega_i^\bullet - \omega_i}{\partial} = f_i(\omega_i, \tau_i), \quad i = 1, 2. \tag{23}$$

At this point, we must identify possible impulsive variables, which is the objective of the following *impulse analysis.*

Impulse analysis. Before engaging the clutch, we must assume $\omega_1 - \omega_2 \neq 0$. Since $\omega_1^\bullet - \omega_2^\bullet = 0$ holds, $\frac{(\omega_1^\bullet - \omega_2^\bullet) - (\omega_1 - \omega_2)}{\partial} = f_1(\omega_1, \tau_1) - f_2(\omega_2, \tau_2)$ cannot be a finite nonstandard real because, if it was, then the function $t \mapsto \omega_1(t) - \omega_2(t)$ would be right-continuous, in contradiction with the above assumption. Hence, the nonstandard real $f_1(\omega_1, \tau_1) - f_2(\omega_2, \tau_2)$ 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 t, and because of equation $(e_4) : \tau_1 + \tau_2 = 0$, both torques are in fact infinite, i.e., are *impulsive.*

Eliminating impulsive variables. It remains to compute the (standard) restart values for the state variables. To this end, we assume that the f_i's are linear in the torques, i.e., each f_i has the following form:

$$f_i(\omega_i, \tau_i) \quad = \quad a_i(\omega_i) + b_i(\omega_i)\tau_i \ , \tag{24}$$

where b_1 and b_2 are the inverse moments of inertia of the rotating masses and a_1 and a_2 are damping factors divided by the corresponding moments of inertia. This 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:

$$\left\{ \begin{array}{ll} \omega_1^\bullet = \omega_1 + \partial.(a_1(\omega_1) + b_1(\omega_1)\tau_1) & (e_1^\partial) \\ \omega_2^\bullet = \omega_2 + \partial.(a_2(\omega_2) + b_2(\omega_2)\tau_2) & (e_2^\partial) \\ \omega_1^\bullet - \omega_2^\bullet = 0 & (e_3^\bullet) \\ \tau_1 + \tau_2 = 0 & (e_4) \end{array} \right. \tag{25}$$

It is tempting to standardize System (25) by simply setting $\partial = 0$ in it. Unfortunately, doing this leaves us with a structurally singular system, since the two torques are then involved in only one equation.

Comment 4 (standardizing functions vs. standardizing equations) To get a better insight into this problem, let us replace for a while the infinitesimal ∂ by a 'small' (standard) time step δ and rewrite System (25) as $F(\omega_1^\bullet, \omega_2^\bullet, \tau_1, \tau_2, \omega_1, \omega_1, \delta) = 0$, where $F{=}0$ stacks the four equations of (25) into one vector equation. Then, the function $\delta \mapsto F(\omega_1^\bullet, \omega_2^\bullet, \tau_1, \tau_2, \omega_1, \omega_1, \delta)$ is continuous in δ at $\delta{=}0$. Consequently, by (12), the standardization of $F(\omega_1^\bullet, \omega_2^\bullet, \tau_1, \tau_2, \omega_1, \omega_1, \partial)$, seen as a function, is obtained by setting $\partial{=}0$ in it.

This, however, becomes incorrect if, instead of the function F, we are interested in the equation $F{=}0$. More precisely, in our example, let Y be the vector that collects the states ω_1, ω_2, and X be the vector that collects the dependent variables $\omega_1^\bullet, \omega_2^\bullet, \tau_1, \tau_2$; then, the function F defined by System (25) is linear in δ, i.e., it has the form $Y \mapsto X$, where X is solution of the algebraic equation

$$[A - \delta{\times}B(Y)]X = C(\delta, Y),$$

in which

$$A = \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{array} \right] \quad ; \quad B = B(Y) = \left[\begin{array}{cccc} 0 & 0 & b_1(\omega_1) & 0 \\ 0 & 0 & 0 & b_2(\omega_2) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right]$$

and vector $C(\delta, Y)$ is appropriately fixed; from now on, we omit Y in the writing. For every $\delta > 0$, the matrix $A{-}\delta{\times}B$ is invertible and we have $X{=}(A{-}\delta{\times}B)^{-1}C$.

If A was nonsingular, we could simply write $X = (I{-}\delta{\times}D)^{-1}A^{-1}C$, where $D = A^{-1}B$. Expanding the inverse around $\delta = 0$ would yield $(I{-}\delta{\times}D)^{-1} = I{+}\delta{\times}D{+}o(\delta)$, implying $X = (I{+}\delta{\times}D)A^{-1}C{+}o(\delta)$. Therefore, the solution $X(\partial)$ of equation $(A{-}\partial{\times}B(Y))X{=}C(\partial, Y)$ would standardize as the solution X of the standard equation $AX = C(0, Y)$. All of this, however, requires that A be nonsingular, which is wrong in our case.[10] □

When A is singular (as in our case), the solution consists in eliminating the impulsive variables from System (25), namely, the two torques. This is performed by symbolic pivoting:

1. Using (e_4) yields $-\tau_2 = \tau_1 =_{\text{def}} \tau$;

[10] This informal reasoning will be formalized in Section 10.2, in the context of nonstandard analysis.

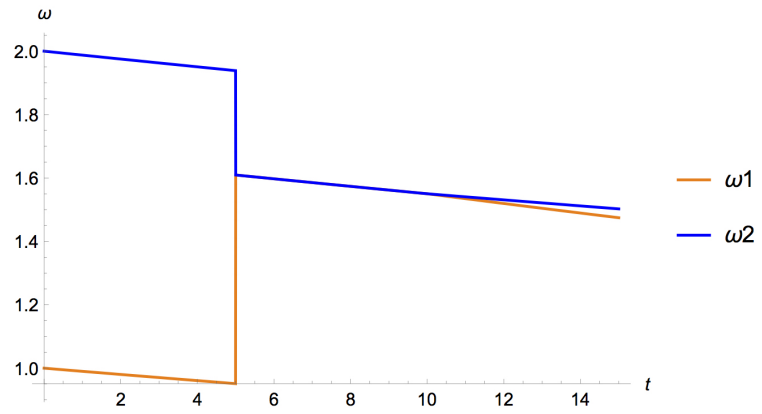


Figure 9: Simulation of the clutch model with resets, using our method. Mode change F → T occurs at $t = 5s$ and mode change T → F occurs at $t = 10s$.

2. Premultiplying the system of equations

$$\begin{cases} \omega_1^\bullet = \omega_1 + \partial.(a_1(\omega_1) + b_1(\omega_1)\tau) & (e_1^\partial) \\ \omega_2^\bullet = \omega_2 + \partial.(a_2(\omega_2) - b_2(\omega_2)\tau) & (e_2^\partial) \end{cases}$$

by the row matrix $\begin{bmatrix} b_2(\omega_2) & b_1(\omega_1) \end{bmatrix}$ yields the equation

$$b_2(\omega_2)\omega_1^\bullet + b_1(\omega_1)\omega_2^\bullet = b_2(\omega_2)(\omega_1 + \partial.a_1(\omega_1)) + b_1(\omega_1)(\omega_2 + \partial.a_2(\omega_2)) \ .$$

3. Using in addition $(e_3^\bullet)$ and setting $\omega^\bullet =_{\text{def}} \omega_1^\bullet = \omega_2^\bullet$ finally yields

$$\omega^\bullet = \frac{b_2(\omega_2)\omega_1 + b_1(\omega_1)\omega_2}{b_1(\omega_1) + b_2(\omega_2)} + \partial \frac{a_1(\omega_1)b_2(\omega_2) + a_2(\omega_2)b_1(\omega_1)}{b_1(\omega_1) + b_2(\omega_2)} \ . \tag{26}$$

The symbolic-only rewriting used here actually alleviates any difficulty in standardizing equation (26): this can now be done by just setting $\partial = 0$ in its right-hand side. This yields, by identifying $\mathsf{st}(\omega_i) = \omega_i^-$ and $\mathsf{st}(\omega_i^\bullet) = \omega_i^+$:

$$\omega_1^+ = \omega_2^+ = \frac{b_2(\omega_2^-)\omega_1^- + b_1(\omega_1^-)\omega_2^-}{b_1(\omega_1^-) + b_2(\omega_2^-)} \ , \tag{27}$$

that is, the reset value for the two velocities is the weighted arithmetic mean of ω_1^- and ω_2^-. Eq. (27) provides us with the reset values for the positions in the engaged mode, which is enough to restart the simulation in this mode. The actual impulsive values for the torques are useless and can be discarded. Note, however, that their sum explicitly remains equal to 0 during the instant of mode change. As a result, the momentum is preserved by the mode change, so that the obtained solution is satisfactory from a physical point of view.

As a final observation, instead of computing the exact standard part of $\omega_i^\bullet$, one could instead attempt to approximate it numerically by substituting ∂ with a small (non-infinitesimal) step size δ in System (25). This alternative approach is developed in Section 11.3. It has the advantage of not requiring the elimination of the impulsive variables.

We thus achieved the objectives stated at the end of Section 1.2.1. Figure 9 shows a simulation of the clutch model where the resets are computed following our approach developed hereabove. As one may expect, the reset value sits between the two values of ω_1^- and ω_2^- when $\gamma : \text{F} \to \text{T}$ (at $t = 5\text{s}$), and the transition is continuous at the second reset (at $t = 10\text{s}$).

3 The Cup-and-Ball example

In this section, we study the example of Section 1.2.2 and address the objectives stated at the end of that section.

3.1 Rejecting the submitted model

We first explain why we reject the model proposed in Section 1.2.2, and how we do so. Let us consider again System (5).

Nonstandard semantics Following Section 2.3 for the clutch, the derivatives x', y', x'', y'' are replaced using the explicit first-order Euler scheme with an infinitesimal time step ∂:

$$x' \leftarrow \frac{x^\bullet - x}{\partial} \quad ; \quad x'' \leftarrow \frac{x^{\bullet 2} - 2x^\bullet + x}{\partial^2} \quad . \tag{28}$$

In the sequel, when referring to model (5), substitution (28) will be implicit.

Structural analysis $x, y, x^\bullet, y^\bullet$ are the state variables, the value of which is known from previous nonstandard instants. The system has guard γ. In contrast to the clutch example, γ is no longer an input, but is rather evaluated based on system variables using equation (k_0). For both modes, the leading variables are $x^{\bullet 2}, y^{\bullet 2}, \lambda$; the first two of these arise from the second derivatives x'', y'' by using substitution (28).

Comment 5 (logico-numerical fixpoints) We could consider the direct solving of System (5), seen as a mixed logico-numerical system of equations with dependent variables $x^{\bullet 2}, y^{\bullet 2}, \lambda, \gamma$. However, this system mixes guarded numerical equations with the evaluation of a predicate. A possible solution would consist in performing a relaxation, by iteratively updating the numerical variables based on the previous value for the guards, and then re-evaluating the guard based on the updated values of the numerical variables, hoping for a fixpoint to occur. Such fixpoint equation, however, can have zero, one, several, or infinitely many solutions. No characterization exists that could serve as a basis for a (graph-based) structural analysis. We thus decided not to try to solve such mixed logico-numerical systems. □

As a consequence, we are unable to evaluate guard γ, so that the mode the system is in cannot be determined. Should we give up? Not yet.

It could be that, based on information obtained by solving the subsystem consisting of the equations that are always enabled, guard γ can be evaluated. In this model, this subsystem is $G = \{(e_1), (e_2)\}$, with dependent variables $x^{\bullet 2}, y^{\bullet 2}, \lambda$. Unfortunately, subsystem G is underdetermined: it cannot be solved, and the evaluation of guard γ is still impossible. We thus reject model (5), because we are unable to evaluate guard γ on the basis of the subsystem that is enabled prior to evaluating this guard.

Comment 6 (forbidding logico-numerical fixpoints based on syntax) The reader may wonder why we did not simply reject the model right from the beginning, on the sole basis of the existence of the logico-numerical fixpoint equation. The rationale is that the method discussed above allows, in principle, more models to get accepted than just the ones without such equations. This design issue will be addressed more thoroughly in Comment 32 of Section 7.1.2. □

3.2 Correcting the model

To break the fixpoint equation defining γ, the idea is to replace the predicate $s \leq 0$ defining the guard γ by its left-limit: $\gamma = [s^- \leq 0]$, where $s^-(t) =_{\text{def}} \lim_{u \nearrow t} s(u)$:

$$\left\{ \begin{array}{rll} & 0 = x'' + \lambda x & (e_1) \\ & 0 = y'' + \lambda y + g & (e_2) \\ & \gamma = [s^- \leq 0] & (k_0) \\ \texttt{if } \gamma \texttt{ then} & 0 = L^2 - (x^2 + y^2) & (k_1) \\ \texttt{and} & 0 = \lambda + s & (k_2) \\ \texttt{if not } \gamma \texttt{ then} & 0 = \lambda & (k_3) \\ \texttt{and} & 0 = (L^2 - (x^2 + y^2)) - s & (k_4) \end{array} \right.$$

We map this model to nonstandard semantics by stating that predicate $s \leq 0$ defines the value of the guard *at the next nonstandard instant*—therefore, an initial condition for guard γ is required; we initialize the system in free motion mode $\gamma(0) = \text{F}$. This yields the corrected model (29), where the modification is highlighted in red:

$$\left\{\begin{array}{rll} & 0 = x'' + \lambda x & (e_1) \\ & 0 = y'' + \lambda y + g & (e_2) \\ & \gamma^\bullet = [s \leq 0]; \gamma(0) = \text{F} & (k_0) \\ \texttt{if } \gamma \texttt{ then} & 0 = L^2 - (x^2 + y^2) & (k_1) \\ \texttt{and} & 0 = \lambda + s & (k_2) \\ \texttt{if not } \gamma \texttt{ then} & 0 = \lambda & (k_3) \\ \texttt{and} & 0 = (L^2 - (x^2 + y^2)) - s & (k_4) \end{array}\right. \tag{29}$$

This model is understood in the nonstandard semantics, meaning that the derivatives are expanded using (28). Therefore, the leading variables in all modes are $\lambda, s, x^{\bullet 2}, y^{\bullet 2}$.

3.3 Nonstandard structural analysis

This section focuses on the core difficulty, namely the mode change $\gamma : \text{F} \to \text{T}$, i.e., the event when the rope gets straight. Due to equation (k_1), the mode $\gamma = \text{T}$ (where the rope is straight) requires index reduction. We thus augment model (29) with the two latent equations shown in red:

$$\left\{\begin{array}{rll} & 0 = x'' + \lambda x & (e_1) \\ & 0 = y'' + \lambda y + g & (e_2) \\ & \gamma^\bullet = [s \leq 0]; \gamma(0) = \text{F} & (k_0) \\ \texttt{if } \gamma \texttt{ then} & 0 = L^2 - (x^2 + y^2) & (k_1) \\ \texttt{and} & 0 = L^2 - (x^2 + y^2)^\bullet & (k_1^\bullet) \\ \texttt{and} & 0 = L^2 - (x^2 + y^2)^{\bullet 2} & (k_1^{\bullet 2}) \\ \texttt{and} & 0 = \lambda + s & (k_2) \\ \texttt{if not } \gamma \texttt{ then} & 0 = \lambda & (k_3) \\ \texttt{and} & 0 = (L^2 - (x^2 + y^2)) - s & (k_4) \end{array}\right. \tag{30}$$

In this model, the two equations (k_1) and $(k_1^\bullet)$ are in conflict with equations from the previous two instants, shown in blue in the following subsystem:

$$\left\{\begin{array}{ll} 0 = \frac{x - 2\,{}^\bullet x + {}^{\bullet 2} x}{\partial^2} + {}^{\bullet 2}\lambda\, {}^{\bullet 2} x & ({}^{\bullet 2} e_1) \\ 0 = \frac{y - 2\,{}^\bullet y + {}^{\bullet 2} y}{\partial^2} + {}^{\bullet 2}\lambda\, {}^{\bullet 2} y + g & ({}^{\bullet 2} e_2) \\ 0 = \frac{x^\bullet - 2x + {}^\bullet x}{\partial^2} + {}^\bullet\lambda\, {}^\bullet x & ({}^\bullet e_1) \\ 0 = \frac{y^\bullet - 2y + {}^\bullet y}{\partial^2} + {}^\bullet\lambda\, {}^\bullet y + g & ({}^\bullet e_2) \\ 0 = L^2 - (x^2 + y^2) & (k_1) \\ 0 = L^2 - (x^2 + y^2)^\bullet & (k_1^\bullet) \end{array}\right.$$

Applying again Principle (1) of causality, we erase, in model (30), equations (k_1) and $(k_1^\bullet)$ at the instant of mode change ${}^\bullet\gamma{=}\text{F}, \gamma{=}\text{T}$. This yields the following system:

$$\text{at} \begin{bmatrix} {}^\bullet\gamma{=}\text{F} \\ \gamma{=}\text{T} \end{bmatrix} : \left\{\begin{array}{ll} 0 = x'' + \lambda x & (e_1) \\ 0 = y'' + \lambda y + g & (e_2) \\ 0 = L^2 - (x^2 + y^2)^{\bullet 2} & (k_1^{\bullet 2}) \\ 0 = \lambda + s & (k_2) \end{array}\right. \tag{31}$$

It uniquely determines all the leading variables from the state variables x, y and $x^\bullet, y^\bullet$. In turn, equations (k_1) and $(k_1^\bullet)$, which were erased from this model, are not satisfied.

Also, we erase, in model (30), the only equation (k_1) at the next instant, i.e., when ${}^{\bullet 2}\gamma{=}\mathrm{F}, {}^{\bullet}\gamma{=}\mathrm{T}, \gamma{=}\mathrm{T}$. This yields the following system:

$$\text{at } \begin{bmatrix} {}^{\bullet 2}\gamma{=}\mathrm{F} \\ {}^{\bullet}\gamma{=}\mathrm{T} \\ \gamma{=}\mathrm{T} \end{bmatrix} : \begin{cases} 0 = x'' + \lambda x & (e_1) \\ 0 = y'' + \lambda y + g & (e_2) \\ 0 = L^2{-}(x^2{+}y^2)^{\bullet} & (k_1^{\bullet}) \\ 0 = L^2{-}(x^2{+}y^2)^{\bullet 2} & (k_1^{\bullet 2}) \\ 0 = \lambda + s & (k_2) \end{cases} \tag{32}$$

This uniquely determines all the leading variables, given values for the state variables x, y and $x^{\bullet}, y^{\bullet}$. Note that $(k_1^{\bullet})$ is a consistency equation that is satisfied by the state variables $x^{\bullet}, y^{\bullet}$. In turn, equation (k_1), which was erased from this model, is not satisfied. At subsequent instants, equation erasure is no longer needed. This completes the nonstandard structural analysis of the mode change $\gamma : \mathrm{F} \to \mathrm{T}$, i.e., when the rope gets straight.

3.4 Standardization

Code generation for restarts consists in standardizing nonstandard systems (31) and (32). Remember, from our development of the clutch example, that standardizing systems of equations requires more care than standardizing numbers, due to impulsive behaviors and singularity issues that result. The simplest method from a conceptual point of view is to eliminate impulsive variables from the restart system, as they are of no use for restarting the new mode.

We focus on the standardization of the mode change $\gamma : \mathrm{F} \to \mathrm{T}$, i.e., when the rope gets straight. Our task is to standardize systems (31) and (32), by targeting discrete-time dynamics, for the two successive instants composing the restart phase. This will provide us with restart values for positions and velocities.

Due to the expansion of derivatives in equations $(e_1, e_2, e_1^{\bullet}, e_2^{\bullet})$, tensions λ and $\lambda^{\bullet}$ are both impulsive, hence so are s and $s^{\bullet}$ by $(k_2, k_2^{\bullet})$. We eliminate the impulsive variables by ignoring $(k_2, k_2^{\bullet})$, combining (e_1) and (e_2) to eliminate λ, and $(e_1^{\bullet})$ and $(e_2^{\bullet})$ to eliminate $\lambda^{\bullet}$. This yields:

$$\text{at } \begin{bmatrix} {}^{\bullet}\gamma{=}\mathrm{F} \\ \gamma{=}\mathrm{T} \end{bmatrix} : \begin{cases} 0 = y''x + gx - x''y \\ 0 = L^2{-}(x^2{+}y^2)^{\bullet 2} \end{cases} \tag{33}$$

$$\text{at } \begin{bmatrix} {}^{\bullet 2}\gamma{=}\mathrm{F} \\ {}^{\bullet}\gamma{=}\mathrm{T} \\ \gamma{=}\mathrm{T} \end{bmatrix} : \begin{cases} 0 = y''x + gx - x''y \\ 0 = L^2{-}(x^2{+}y^2)^{\bullet} \\ 0 = L^2{-}(x^2{+}y^2)^{\bullet 2} \end{cases} \tag{34}$$

In System (33), we expand second derivatives using (28). Note that substitution (28) implies that we can as well expand the second derivatives as functions of first derivatives:

$$x'' \quad \leftarrow \quad \frac{x'^{\bullet} - x'}{\partial} \ . \tag{35}$$

Since we will use System (34) to evaluate restart values for the velocities, we will use expansion (35) in it. Consequently, System (33) has dependent variables $x^{\bullet 2}, y^{\bullet 2}$, whereas System (34) has dependent variables $x'^{\bullet}, y'^{\bullet}$. We are now ready to standardize the two systems.

System (33) to define restart positions We expand second derivatives using (28):

$$\begin{cases} 0 = (y^{\bullet 2} - 2y^{\bullet} + y)x - (x^{\bullet 2} - 2x^{\bullet} + x)y + \partial^2 gx \\ 0 = L^2{-}(x^2{+}y^2)^{\bullet 2} \end{cases} \tag{36}$$

Regarding (36), one can safely set $\partial = 0$ to standardize it, since the resulting system is still structurally regular—see Comment 4 regarding the clutch example. The standardization of (36) is thus:

$$\begin{cases} 0 = (y^{\bullet 2} - 2y^{\bullet} + y)x - (x^{\bullet 2} - 2x^{\bullet} + x)y \\ 0 = L^2{-}(x^2{+}y^2)^{\bullet 2} \end{cases} \tag{37}$$

In contrast, had we directly set $\partial = 0$ in System (31) (without eliminating impulsive variable λ), we would get a structurally singular system, an incorrect standardization.

In System (37), we can interpret x and $x^\bullet$ as the left-limit x^- of state variable x in previous mode, and $x^{\bullet 2}$ as the restart value x^+ for the new mode. This yields

$$\left\{ \begin{array}{l} 0 = (y^+ - y^-)x^- - (x^+ - x^-)y^- \\ 0 = L^2 - (x^2+y^2)^+ \end{array} \right. \tag{38}$$

which determines the restart values for positions. Note that the constraint that the rope is straight is satisfied. Furthermore, if we remember that the rope is straight at mode change (this is the mode change condition), then $0 = L^2 - (x^2+y^2)^-$ also holds, which ensures that $x^+ = x^-, y^+ = y^-$ is the unique solution of (38): positions are continuous.

System (34) to define restart velocities We expand second derivatives using (35):

$$\left\{ \begin{array}{l} 0 = (y'^\bullet - y')x - (x'^\bullet - x')y + \partial.gx \\ 0 = L^2 - (x^2+y^2)^\bullet \\ 0 = L^2 - (x^2+y^2)^{\bullet 2} \end{array} \right. \tag{39}$$

By expanding $x^{\bullet 2} = x^\bullet + \partial x'^\bullet$, the right-hand side of the last equation rewrites

$$\begin{array}{rcl} L^2 - (x^2+y^2)^{\bullet 2} = & & L^2 - (x^2+y^2)^\bullet \\ & + & 2\partial(x^\bullet x'^\bullet + y^\bullet y'^\bullet) \\ & + & \partial^2\big((x'^\bullet)^2 + (y'^\bullet)^2\big) \\ = & & 0 \ \text{(by second equation of (39))} \\ & + & 2\partial(x^\bullet x'^\bullet + y^\bullet y'^\bullet) \\ & + & O(\partial^2) \end{array} \tag{40}$$

Using this expansion of $L^2 - (x^2+y^2)^{\bullet 2}$, setting $\partial = 0$ in (39) yields

$$\left\{ \begin{array}{l} 0 = (y'^\bullet - y')x - (x'^\bullet - x')y \\ 0 = x^\bullet x'^\bullet + y^\bullet y'^\bullet \end{array} \right. \tag{41}$$

where the dependent variables are now $x'^\bullet, y'^\bullet$, whereas other variables are state variables whose values are determined by previous time steps. Note that System (41) is structurally regular, hence it is the correct standardization of System (39). We are now ready to get restart values for velocities. In System (41), we perform the following substitutions, recalling that superscripts $^-$ and $^+$ denote left- and right-limits:

$$x = x^- \ ; \ x^\bullet = x^+ \ ; \ x' = x'^- \tag{42}$$

and the restart velocity is evaluated by

$$x'^+ = x'^\bullet \quad . \tag{43}$$

After similar substitutions for y, we get

$$\left\{ \begin{array}{l} 0 = (y'^+ - y'^-)x^- - (x'^+ - x'^-)y^- \\ 0 = x^+ x'^+ + y^+ y'^+ \end{array} \right. \tag{44}$$

System (44) determines x'^+ and y'^+, which are the velocities for restart. The second equation guarantees that the velocity will be tangent to the constraint. With (38) and (44), we determine the restart conditions for positions and velocities. Invariants from the physics are satisfied.

Our reasoning so far produces a behavior in which the two modes (free motion and straight rope) gently alternate; the system always stays in one mode for some positive period of time before switching to the other mode. *This indeed amounts to assuming that the impact is totally inelastic at mode change, an assumption that was not explicit at all in* (29). So, what happened?

3.5 Handling transient modes

In fact, *the straight rope mode was implicitly assumed to last for at least three nonstandard successive instants, since we allowed ourselves to shift* (k_1) *twice.*

Now, let us instead assume elastic impact. Hence, the cascade of mode changes $\gamma : \text{F} \to \text{T} \to \text{F}$ is possible in three successive nonstandard instants, reflecting that the straight rope mode is *transient* (it is left immediately after being reached). We wish to adapt our reasoning to handle this transient mode.

Consider again model (29). We regard the instant of the cascade when $\gamma = \text{T}$ occurs as the current instant. We cannot add latent equations by simply shifting (k_1), since these shifted versions are not active in the mode $\gamma = \text{F}$. Instead, our line of reasoning goes as follows. Set

$$\begin{array}{rcl} S(\text{T}) & = & \{(e_1),(e_2),(k_1),(k_2)\} \ , \\ S(\text{F}) & = & \{(e_1),(e_2),(k_3),(k_4)\} \ . \end{array}$$

Systems $S^\bullet(\text{T})$ and $S^\bullet(\text{F})$ are obtained by shifting the equations constituting $S(\text{T})$ and $S(\text{F})$; systems $S^{\bullet 2}(\text{T})$ and $S^{\bullet 2}(\text{F})$ are defined similarly.

Now, the idea is to consider the *differentiation array* originally proposed by Campbell and Gear [21], except that we take into account the trajectory $\text{T}, \text{F}, \text{F}, \ldots$ for guard γ. Using shifting instead of differentiation yields the following *array of shifted systems*, where size n is a parameter:

$$\mathcal{A}_n(S) \quad =_{\text{def}} \quad \left[\begin{array}{c} S(\text{T}) \\ S^\bullet(\text{F}) \\ S^{\bullet 2}(\text{F}) \\ \vdots \\ S^{\bullet n}(\text{F}) \end{array}\right] \tag{45}$$

We search for an n such that $\mathcal{A}_n(S)$ provides us with the due latent equations. Thus, the question is: can we find latent equations for $S(\text{T})$ by shifting appropriate equations from $S(\text{F})$? Unfortunately, the answer is no: although shifting twice (k_4) in System (29) produces one more equation involving the leading variables $x^{\bullet 2}, y^{\bullet 2}$, this equation also involves the new variable $s^{\bullet 2}$, which keeps the augmented system underdetermined; shifting other equations fails as well.

Therefore, the structural analysis rejects this model as being underdetermined at transient mode $\gamma = \text{T}$. The user is then asked to provide one more equation. For example, he/she could specify an impact law for the velocity y' by providing the equation

$$(y')^+ = -(1-\alpha)(y')^- \ ,$$

where $0 \leq \alpha < 1$ is a fixed damping coefficient. This is reinterpreted in the nonstandard domain as

$$y'^\bullet = -(1-\alpha)y' \ , \tag{46}$$

yielding the following refined system for use at mode $\gamma{=}\text{T}$ within the cascade $\gamma{:}\text{F}{\to}\text{T}{\to}\text{F}$:

$$\left\{\begin{array}{ll} 0 = x'' + \lambda x & (e_1) \\ 0 = y'' + \lambda y + g & (e_2) \\ 0 = y'^\bullet + (1-\alpha)y' & (\tau_1) \\ 0 = L^2 - (x^2+y^2) & (k_1) \\ 0 = \lambda + s & (k_2) \end{array}\right. \tag{47}$$

We proceed again with the structural analysis. Variables x, y are the states, so that their values are set by the previous instants. Current equation (k_1) creates a conflict with the past. Hence, we discard it from System (47), which leaves us with the following system:

$$\left\{\begin{array}{ll} 0 = x'' + \lambda x & (e_1) \\ 0 = y'' + \lambda y + g & (e_2) \\ 0 = y'^\bullet + (1-\alpha)y' & (\tau_1) \\ 0 = \lambda + s & (k_2) \end{array}\right. \tag{48}$$

Model (48) is structurally nonsingular, recalling that y'' and $y'^{\bullet}$ can be interchanged for the structural analysis. This refined model is therefore accepted. Standardization proceeds accordingly, with no new difficulty.

3.6 Consequences for the modeling language

Through the Cup-and-Ball example, we demonstrated the need for the following user-given information: is the current mode long or transient? We argue that this information cannot be derived from inspecting the system model.

In the clutch example, guard γ was a system input, and we implicitly assumed that each of the two modes was long. Had we composed the clutch model with a 'feedback' determining γ as a predicate over system variables, this implicit assumption would have become questionable. This situation indeed occurs with the Cup-and-Ball example, where the elastic/inelastic impact assumption is a side information that cannot be deduced from the equation syntax. This leads to the following important observation:

Comment 7 (Long / Transient) Long / Transient is an information regarding modes, that cannot be found by inspecting the equations of the system. It must be inferred from understanding the system physics. Therefore, it has to be provided as an extra information by the model designer. □

The natural way of addressing Comment 7 is to provide a different syntax for specifying long modes on the one hand, and events corresponding to transient modes on the other hand. In contrast, mode changes separating two successive long modes need not be specified. A candidate syntax is:

long mode	`if predicate then` S
transient mode	`when predicate then` S

(49)

The '`when`' statement of Modelica, which we reuse here, points to the event when '`predicate`' switches from false to true. For this '`when`' statement, we could furthermore restrict '`predicate`' to be a zero-crossing condition, by which a real-valued expression crosses zero from below [15]. We devote the '`if`' statement to long-lasting modes specified by '`predicate`'.

To illustrate this, our syntax offering `if`-equations and `when`-equations allows specifying the Cup-and-Ball example with elastic impact as shown in Figure 10.

$$\left\{\begin{array}{rll} & 0 = x'' + \lambda x & (e_1)\\ & 0 = y'' + \lambda y + g & (e_2)\\ & \gamma = [s^- \leq 0]; \gamma(0) = \text{F} & (k_0)\\ \texttt{when } \gamma \texttt{ then} & y' = -\alpha y'^- & (\tau_1)\\ \texttt{if not } \gamma \texttt{ then} & 0 = \lambda & (k_3)\\ \texttt{and} & 0 = (L^2 - (x^2+y^2)) - s & (k_4) \end{array}\right.$$

Figure 10: Model of the Cup-and-Ball example with elastic impact specified by equation (τ_1), highlighted in red. This equation replaces the two equations $(k_1), (k_2)$ specifying the straight rope mode in case of inelastic impact.

4 The Westinghouse air brake example

In this section we develop the example of Section 1.2.3 and we address the objectives stated at the end of that section.

We reject the model proposed in Section 1.2.3 for the very same reasons as for the Cup-and-Ball example, namely the existence of a logico-numerical fixpoint equation defining the guard. The corrected system is shown in Figure 11. The two systems differ in equation (v_0), where the predicate

$$
\begin{array}{rll}
 & 0 = f_b - f_v - f_{cl} + f_l & (m_1) \\
 & 0 = f_v - f_{ch} - f_l - f_t & (m_2) \\
 & 0 = f_b - F_1.\varphi(p_b - p_r) & (f_1) \\
 & 0 = F_1.\varphi(p_{bn} - p_r) & (f_2) \\
 & 0 = f_l - F_2.\varphi(p_t - p_r) & (f_3) \\
 & 0 = f_t - \frac{\rho.V}{P_0}.p'_t & (r) \\
 & 0 = \rho.S.(x'.p_r + (x - L).p'_r) + P_0.f_{cl} & (p_1) \\
 & 0 = \rho.S.(x'.S.p_t + x.p'_t) - P_0.f_{ch} & (p_2) \\
 & 0 = S.(p_t - p_r) - b & (l_1) \\
 & 0 = K.x - b & (l_2) \\
 & \gamma^\bullet = [s \leq 0]; \gamma(0) = \text{F} & (v_0) \\
\texttt{if } \gamma \texttt{ then} & 0 = p_r - p_t & (v_1) \\
\texttt{and} & 0 = f_v + s & (v_2) \\
\texttt{if not } \gamma \texttt{ then} & 0 = f_v & (v_3) \\
\texttt{and} & 0 = p_r - p_t - s & (v_4)
\end{array}
$$

Figure 11: The system of of Figure 8 corrected. The correction is highlighted in red.

$s < 0$ sets the value of guard γ for the next nonstandard instant. This models the fact that γ is evaluated based on ${}^\bullet s$, which is the nonstandard semantics of the left-limit of s. Note that an initial condition for γ is needed in this corrected version. This model is interpreted in the nonstandard semantics, meaning that substitution (28) applies.

4.1 Nonstandard structural analysis

The value of guard γ is known from the previous instant, together with the value for the two state variables x, p_r, p_t. For both modes $\gamma = \text{T/F}$, the leading variables for evaluation are the 12 variables $x^\bullet, p_r^\bullet, p_t^\bullet, f_b, f_v, f_{cl}, f_l, f_{ch}, f_t, p_{bn}, b, s$, hence the active system is square in both modes. We successively explore the two long modes $\gamma = \text{T/F}$ for the guard. Then we investigate the mode changes.

Long mode $\gamma = \text{F}$. The active system is $G_\text{F} = G \cup \{(v_3), (v_4)\}$, see Figure 12.

The subsystem $\{(l_1), (l_2)\}$ involves two equations but only one leading variable: b. It is structurally singular. Following the technique of index reduction used in the clutch example, we augment G_F with the latent equations $(l_1^\bullet), (l_2^\bullet)$. The result is shown in Figure 13. Note that b is now a state variable. Equations $(l_1), (l_2)$ are consistency conditions. They are satisfied as a result of executing the previous instant. We are thus left with the system $G_\text{F}^\Sigma = G \cup \{(l_1^\bullet), (l_2^\bullet)\}$. Can we check its regularity by using structural (graph based) methods?

A *complete matching* is shown in red in Figure 13, i.e., a one-to-one assignment of leading variables to equations. The intent is that each variable is evaluated using its assigned equation, by pivoting. For example, f_{cl} is evaluated using equation (m_1). The heart of structural analysis for algebraic systems of equations is that, if a complete matching exists, then the system is regular in a generic sense, i.e., outside exceptional values for the non-zero coefficients of the equations. See Section 6 for a formal development.

The system G_F^Σ is thus structurally nonsingular. Hence, the system of Figure 12 was of index 1 and adding the latent equations reduced the index to 0.

Long mode γ=T. The active system is $G_\text{T} = G \cup \{(v_1), (v_2)\}$, shown in Figure 14. The subsystem $\{(l_1), (l_2), (v_1)\}$ is structurally singular since it has 3 equations and only one dependent variable b. We thus shift it forward. The result is shown in Figure 15.

Equations $(l_1), (l_2), (v_1)$ are consistency conditions. They are satisfied as a result of executing the previous instant. We are thus left with the index reduced system $G_\text{T}^\Sigma = G \cup \{(l_1^\bullet), (l_2^\bullet), (v_1^\bullet)\}$, for which a complete matching is shown in red in Figure 15. This system is structurally nonsingular.

$$
\begin{array}{ll}
0 = f_b - f_v - f_{cl} + f_l & (m_1) \\
0 = f_v - f_{ch} - f_l - f_t & (m_2) \\
0 = f_b - F_1.\varphi(p_b - p_r) & (f_1) \\
0 = F_1.\varphi(p_{bn} - p_r) & (f_2) \\
0 = f_l - F_2.\varphi(p_t - p_r) & (f_3) \\
0 = f_t - \frac{\rho.V}{P_0}.p'_t & (r) \\
0 = \rho.S.(x'.p_r + (x - L).p'_r) + P_0.f_{cl} & (p_1) \\
0 = \rho.S.(x'.S.p_t + x.p'_t) - P_0.f_{ch} & (p_2) \\
0 = S.(p_t - p_r) - b & (l_1) \\
0 = K.x - b & (l_2) \\
0 = f_v & (v_3) \\
0 = p_r - p_t - s & (v_4)
\end{array}
$$

Figure 12: The active system when $\gamma = \text{F}$. We show in red a structurally singular subsystem.

$$
\begin{array}{ll}
0 = f_b - f_v - f_{cl} + f_l & (m_1) \\
0 = f_v - f_{ch} - f_l - f_t & (m_2) \\
0 = f_b - F_1.\varphi(p_b - p_r) & (f_1) \\
0 = F_1.\varphi(p_{bn} - p_r) & (f_2) \\
0 = f_l - F_2.\varphi(p_t - p_r) & (f_3) \\
0 = f_t - \frac{\rho.V}{P_0}.p'_t & (r) \\
0 = \rho.S.(x'.p_r + (x - L).p'_r) + P_0.f_{cl} & (p_1) \\
0 = \rho.S.(x'.S.p_t + x.p'_t) - P_0.f_{ch} & (p_2) \\
0 = S.(p_t^\bullet - p_r^\bullet) - b^\bullet & (l_1^\bullet) \\
0 = K.x^\bullet - b^\bullet & (l_2^\bullet) \\
0 = f_v & (v_3) \\
0 = p_r - p_t - s & (v_4) \\
\text{consistency equations:} & \\
0 = S.(p_t - p_r) - b & (l_1) \\
0 = K.x - b & (l_2)
\end{array}
$$

Figure 13: The system of Figure 12 in which latent equations are found by shifting $(l_1), (l_2)$. Recall that x' and $x^\bullet$ are related by (28). The assignment of a variable to each equation, highlighted in red, defines a complete matching.

The system of Figure 14 was of index 1. The set of latent equations differs from that of the other mode, however.

So far we have completed the study of the long modes. It remains to investigate the mode changes.

Mode change $\gamma : \text{T} \to \text{F}$. We reproduce the reasoning following System (20) for the clutch example. Joining together the systems at previous and current instant may result in a conflict between the consistency equations of the current instant and selected equations from the previous instant. This does not occur here, however: consistency equations $(l_1), (l_2)$ in Figure 13 coincide with the latent equations $(l_1^\bullet), (l_2^\bullet)$ associated with the previous instant in Figure 15 (the set of latent equations of mode $\gamma = \text{T}$ contains the set of latent equations of mode $\gamma = \text{F}$).

Mode change $\gamma : \text{F} \to \text{T}$. Unlike for the previous case, joining together the systems at previous and current instant actually results in a conflict between the consistency equations of the current instant and the dynamics at the previous instant. More precisely, the consistency equation $(v_1) : 0{=}p_r - p_t$, which holds in current mode $\gamma = \text{T}$, is not a consequence of executing the previous instant (in mode

$$
\begin{array}{ll}
0 = f_b - f_v - f_{cl} + f_l & (m_1) \\
0 = f_v - f_{ch} - f_l - f_t & (m_2) \\
0 = f_b - F_1.\varphi(p_b - p_r) & (f_1) \\
0 = F_1.\varphi(p_{bn} - p_r) & (f_2) \\
0 = f_l - F_2.\varphi(p_t - p_r) & (f_3) \\
0 = f_t - \frac{\rho.V}{P_0}.p'_t & (r) \\
0 = \rho.S.(x'.p_r + (x - L).p'_r) + P_0.f_{cl} & (p_1) \\
0 = \rho.S.(x'.S.p_t + x.p'_t) - P_0.f_{ch} & (p_2) \\
0 = S.(p_t - p_r) - b & (l_1) \\
0 = K.x - b & (l_2) \\
0 = p_r - p_t & (v_1) \\
0 = f_v + s & (v_2)
\end{array}
$$

Figure 14: The active system when $\gamma = \text{T}$. We show in red a structurally singular subsystem. Compare with Figure 12.

$$
\begin{array}{ll}
0 = f_b - f_v - f_{cl} + f_l & (m_1) \\
0 = f_v - f_{ch} - f_l - f_t & (m_2) \\
0 = f_b - F_1.\varphi(p_b - p_r) & (f_1) \\
0 = F_1.\varphi(p_{bn} - p_r) & (f_2) \\
0 = f_l - F_2.\varphi(p_t - p_r) & (f_3) \\
0 = f_t - \frac{\rho.V}{P_0}.p'_t & (r) \\
0 = \rho.S.(x'.p_r + (x - L).p'_r) + P_0.f_{cl} & (p_1) \\
0 = \rho.S.(x'.S.p_t + x.p'_t) - P_0.f_{ch} & (p_2) \\
0 = S.(p_t^\bullet - p_r^\bullet) - b^\bullet & (l_1^\bullet) \\
0 = K.x^\bullet - b^\bullet & (l_2^\bullet) \\
0 = p_r^\bullet - p_t^\bullet & (v_1^\bullet) \\
0 = f_v + s & (v_2) \\
\text{consistency equations:} & \\
0 = S.(p_t - p_r) - b & (l_1) \\
0 = K.x - b & (l_2) \\
0 = p_r - p_t & (v_1)
\end{array}
$$

Figure 15: Adding latent equations to the system of Figure 14.

$\gamma = \text{F}$). As for the clutch example (Section 2.4 and particularly (1)),

$$
\begin{array}{l}
\text{we remove, at the instant of mode change } \gamma : \text{F} \to \text{T}, \\
\text{the consistency equation } (v_1) : 0{=}p_r - p_t.
\end{array}
\tag{50}
$$

This completes the structural analysis.

4.2 Standardization

To derive the actual executable code we must return to the standard domain by performing standardization. The task is very much like for the clutch example and we only provide a brief summary:

Within long modes $\gamma = \text{F}$ or T: Here, time and dynamics are continuous: We standardize $\frac{x^\bullet - x}{\partial}$ as the derivative x'. Also, we standardize equation (v_0) as $\gamma{=}[s^- {\leq} 0]$, where s^- denotes the left limit of s, i.e., the value of s before the mode change in case a mode change occurs. Doing this removes all the occurrences of ∂ in both modes. The resulting system is standard.

For mode changes: Here we target discrete time: $x^\bullet$ is interpreted as the next value for x and raises no difficulty. In contrast, the occurrence of ∂ in the Euler schemes $\frac{x^\bullet - x}{\partial}$, where it acts in space, is now the difficulty.

Mode change $\gamma : \text{T} \to \text{F}$. The restart conditions are solutions of the system of Figure 13 in which the consistency equations are ignored. This system coincides with the nonstandard dynamics of the long mode $\gamma = \text{F}$. However, we do not seek for a strandardization in continuous time, but rather in discrete time, which is different: derivatives are expanded using (28) and we solve the system for the discrete time leading variables $p_t^\bullet, p_r^\bullet, x^\bullet$, etc. This nonstandard system is shown in Figure 16, where the infinitesimal ∂ is highlighted in blue.

$$
\begin{array}{ll}
0 = f_b - f_v - f_{cl} + f_l & (m_1) \\
0 = f_v - f_{ch} - f_l - f_t & (m_2) \\
0 = f_b - F_1.\varphi(p_b - p_r) & (f_1) \\
0 = F_1.\varphi(p_{bn} - p_r) & (f_2) \\
0 = f_l - F_2.\varphi(p_t - p_r) & (f_3) \\
0 = \partial.f_t - \frac{\rho.V}{P_0}.(p_t^\bullet - p_t) & (r) \\
0 = \rho.S.((x^\bullet - x).p_r + (x - L).(p_r^\bullet - p_r)) + \partial.P_0.f_{cl} & (p_1) \\
0 = \rho.S.((x^\bullet - x).S.p_t + x.(p_t^\bullet - p_t)) - \partial.P_0.f_{ch} & (p_2) \\
0 = S.(p_t^\bullet - p_r^\bullet) - b^\bullet & (l_1^\bullet) \\
0 = K.x^\bullet - b^\bullet & (l_2^\bullet) \\
0 = f_v & (v_3) \\
0 = p_r - p_t - s & (v_4)
\end{array}
$$

Figure 16: Restart at mode change $\gamma : \text{T} \to \text{F}$.

We need to standardize this system. We leave to the reader as an exercise that setting $\partial = 0$ in this system makes it structurally nonsingular (no complete matching can be found). By Comment 4 of Section 2.5.2, this implies that setting $\partial = 0$ in this system does not lead to a correct standardization.

The problem originates from equations $(r), (p_1), (p_2)$. Focus on (r): the state variable x is finite before and after the mode change and $x^\bullet - x \neq 0$ possibly holds; then, ∂ infinitesimal requires that f_t is possibly infinite, i.e., impulsive. The same holds for f_{cl} and f_{ch}.

Following the approach of Section 2.5.2, we must eliminate, by algebraic manipulations, the possibly impulsive variables f_t, f_{cl}, f_{ch}. This is easy since the latter enter linearly in equations $(r), (p_1), (p_2)$.

Setting $\partial = 0$ in the resulting system leaves us with a structurally singular system that sets the restart values for the system state variables p_r, p_t, b, x. This yields the desired standardization, which is the code for correct restart at this mode change. Details are omitted.

Mode change $\gamma : \text{F} \to \text{T}$. We proceed similarly. The restart equations at mode change $\gamma : \text{F} \to \text{T}$ are given in Figure 17, where the infinitesimal ∂ is highlighted in blue. We conclude as for the other

$$
\begin{array}{ll}
0 = f_b - f_v - f_{cl} + f_l & (m_1) \\
0 = f_v - f_{ch} - f_l - f_t & (m_2) \\
0 = f_b - F_1.\varphi(p_b - p_r) & (f_1) \\
0 = F_1.\varphi(p_{bn} - p_r) & (f_2) \\
0 = f_l - F_2.\varphi(p_t - p_r) & (f_3) \\
0 = \partial.f_t - \frac{\rho.V}{P_0}.(p_t^\bullet - p_t) & (r) \\
0 = \rho.S.((x^\bullet - x).p_r + (x - L).(p_r^\bullet - p_r)) + \partial.P_0.f_{cl} & (p_1) \\
0 = \rho.S.((x^\bullet - x).S.p_t + x.(p_t^\bullet - p_t)) - \partial.P_0.f_{ch} & (p_2) \\
0 = S.(p_t^\bullet - p_r^\bullet) - b^\bullet & (l_1^\bullet) \\
0 = K.x^\bullet - b^\bullet & (l_2^\bullet) \\
0 = p_r^\bullet - p_t^\bullet & (v_1^\bullet) \\
0 = f_v + s & (v_2)
\end{array}
$$

Figure 17: Restart at mode change $\gamma : \text{F} \to \text{T}$.

mode change.

Comment 8 (fake impulsive variables) The reader acquainted with the physics of this system may be quite surprised by our reasoning above. We found that, for the two mode changes, algebraic variables f_t, f_{cl}, and f_{ch} are possibly impulsive. However, the physics of the Westinghouse air brake does not tell so. There seems to be something wrong here. □

In the next section we investigate how the right solution can be found at compile time.

4.3 Assertions

Focus on the system for restart at mode change $\gamma : \text{F} \to \text{T}$, see Figure 17 and assume for a while that the trajectories of system variables remain continuous, which applies in particular to s. Now, the predicate defining the guard γ tells us that the value of s at both mode changes is zero. As a consequence, at the instant of both mode changes $\gamma : \text{F} \to \text{T}$ and $\text{T} \to \text{F}$, the two subsystems $\{(v_1), (v_2)\}$ and $\{(v_3), (v_4)\}$ of the system of Figure 11 become identical due to the zero value for s. As a consequence, whatever next mode is, the consistency equations for it ($\{(v_1), (v_2)\}$ or $\{(v_3), (v_4)\}$) will be satisfied. Therefore, no action is required at mode changes other than commuting from code of Figure 13 to code of Figure 15 or vice-versa.

Since our compiler is only equipped with our graph based structural analysis, it will not be able to perform the above reasoning. We can, however, overcome this difficulty by giving to the user the possibility to manually refine her model by adding *guarded assertions* of the form

$$\texttt{when}\ \gamma\ \texttt{assert}\ \beta(X) \tag{51}$$

where γ is some guard and $\beta(X)$ is a predicate in the numerical variables X of the system. For example, in the Westinghouse example, the user could state the following assertion, which follows from the fact that $s = 0$ holds at mode changes:

$$\texttt{when}\ \gamma^{-}\ \texttt{and not}\ \gamma\ \texttt{assert}\ s = 0 \tag{52}$$

Assertion (52) can be used by the compiler in deducing that no specific restart action is required other than commuting between the code of the two modes, which, in turn, implies that no state variable is impulsive.

Comment 9 (are assertions needed for correctness?) Not quite. In fact, the approach followed in Section 4.2 for deriving the restart conditions at mode changes is still correct. It is just incomplete in that it does not allow to compute the value for the algebraic variables that were erroneously considered impulsive. This is indeed harmless. Still, our approach of Section 4.2 was an overshoot and adding a proper assertion is preferred. Also, assertions can be very useful in narrowing the domain of reachable modes, which may improve efficiency of the compilation. □

Assertions are provided in our IsamDAE tool under development [19, 20], see also Section 12. We will not discuss them any further in this paper, however.

5 Issues and systematic approach

From our discussion of the examples, we see that the key issue is the handling of mode changes, and particularly the reconciliation of the dynamics just before the change, and the consistency conditions set by the new mode. How to properly address this in the original model of continuous real-time was an open question; mapping the system dynamics to the nonstandard domain clarified this task.

In the rest of this paper, we extend our method, from the toy examples to a systematic approach for mDAE system compilation. This approach, illustrated in Figure 18, decomposes into several steps.

- First, the mDAE system is transformed into a system of *multi-mode difference Algebraic Equations* (mdAE) using the nonstandard Euler expansion of derivatives $x' \leftarrow \frac{x^\bullet - x}{\partial}$. The resulting mdAE system is the *nonstandard semantics* of the original model.
- Second, the structural analysis of this nonstandard semantics is performed, resulting in a new mdAE system where latent equations are made explicit and conflicts at mode changes are solved.
- Finally, the standardization step recovers both the smooth dynamics in each mode, and the possibly discontinuous/impulsive restart conditions at mode changes. The resulting code can then be executed with the help of a solver.

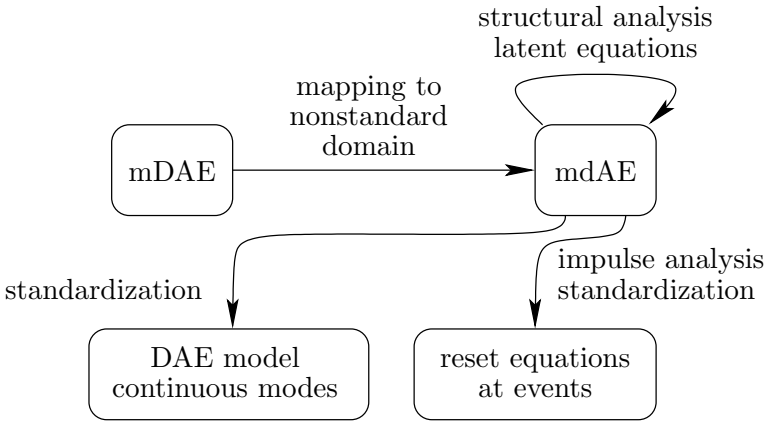


Figure 18: Our approach to the compilation of mDAE systems.

The intuitions developed on the examples are formalized in the sequel of the paper. We first recall the background on structural analysis and we extend it to our context. We then focus on standardization.

6 Background on Structural Analysis

We recall in this section the basics of structural analysis for algebraic systems of equations and provide in passing a few new lemmas for subsequent use. We then present the Σ-method, which is the structural analysis method we adopt for DAE systems.

Notations 10 In this paper, $\mathbb{N} = \{0, 1, 2, \dots\}$ is the set of nonnegative natural integers and $\mathbb{R}_+ = [0, \infty)$ is the set of nonnegative real numbers. □

We will sometimes need to handle variables, their valuations, and vectors thereof. To this end, the following conventions will be used (unless no confusion occurs from using more straightforward notations):

Notations 11 Lowercase letters (x, y, etc.) denote scalar real variables; capitals (X, Y, etc.) denote vectors of real variables; whenever convenient, we regard X as a set of variables and write $x \in X$ to refer to an entry of X. We adopt similar conventions for functions (f, g, etc.) and vectors of functions (F, G, etc.). A value for a variable x is generically denoted by ν_x, and similarly for X and ν_X. □

6.1 Structural analysis of algebraic equations

In this section we focus on systems of algebraic equations, that is, equations involving no time and no dynamics.

6.1.1 Structural nonsingularity of algebraic equations

Consider a system of smooth algebraic equations:

$$f_i(y_1, \ldots, y_k, x_1, \ldots, x_n) = 0, \quad i = 1, \ldots, m \tag{53}$$

rewritten as $F(Y, X) = 0$, where Y and X denote the vectors $(y_1, \ldots, y_k)$ and $(x_1, \ldots, x_n)$, respectively, and F is the vector $(f_1, \ldots, f_m)$. The system has m equations, k input variables collected in vector Y, and n *dependent variables* (or unknowns) collected in vector X. Throughout this section , we assume that the f_i's are all $\mathcal{C}^1$ at least.

If the system is square, i.e., if $m = n$, the *Implicit Function Theorem* (see, e.g., Theorem 10.2.2 in [24]) states that, if $(\nu_Y, \nu_X) \in \mathbb{R}^{k+n}$ is a value for the pair (Y, X) such that $F(\nu_Y, \nu_X) = 0$ and the Jacobian matrix of F with respect to X evaluated at (ν_Y, ν_X) is nonsingular, then there exists, in an open neighborhood U of ν_Y, a unique vector of functions G such that $F(v, G(v)) = 0$ for all $v \in U$. In words, Eq. (53) uniquely determines X as a function of Y, locally around ν_Y. Denote by $\mathbf{J}_X F$ the above mentioned Jacobian matrix. Solving $F = 0$ for X, given a value ν_Y for Y, requires forming $\mathbf{J}_X F(\nu_Y)$ as well as inverting it.

One could avoid considering $\mathbf{J}_X F$ by focusing on *structural* nonsingularity, a weaker but much cheaper criterion previously introduced for handling sparse matrices in high performance computing.

6.1.2 Structural analysis

We begin with some background on graphs, for which a basic reference is [11]. Given a graph $\mathcal{G} = (V, E)$, where V and E are the sets of vertices and edges, a *matching* $\mathcal{M}$ of $\mathcal{G}$ is a set of edges of $\mathcal{G}$ such that no two edges in $\mathcal{M}$ are incident on a common vertex. Matching $\mathcal{M}$ is called *complete* if it covers all the vertices of $\mathcal{G}$. An *$\mathcal{M}$-alternating path* is a path of $\mathcal{G}$ whose edges are alternatively in $\mathcal{M}$ and not in $\mathcal{M}$—we say simply "alternating path" when $\mathcal{M}$ is understood from the context. A vertex is said to be *matched* if there is an edge in $\mathcal{M}$ that is incident on the vertex, and *unmatched* otherwise.

Let $F(Y, X)$=0 be a system of algebraic equations of the form (53); its *bipartite graph* $\mathcal{G}_F$ is the graph having $F \cup X$ as set of vertices and an edge (f_i, x_j) if and only if variable x_j occurs in function f_i.

Square systems We first consider square systems, in which equations and dependent variables are in equal numbers.

Definition 12 (structural nonsingularity) *System $F(Y, X)$=0 is called* structurally nonsingular *if its bipartite graph $\mathcal{G}_F$ possesses a complete matching.*

A structurally nonsingular system is necessarily square, i.e., with equations and variables in equal numbers. The link to numerical regularity is formalized in the following lemma (implicitly used in [40, 37, 46, 48]):

Lemma 13 *Assume that F is $\mathcal{C}^1$. The following properties are equivalent:*

1. *System $F(Y, X)$=0 is structurally nonsingular.*
2. *For every (ν_Y, ν_X) satisfying $F(\nu_Y, \nu_X)$=0, the Jacobian matrix $\mathbf{J}_X F(\nu_Y, \nu_X)$ remains generically*[11] *nonsingular when its nonzero coefficients vary over some neighborhood.*

Let us detail Property 2. We can represent graph $\mathcal{G}_F$ by the $n \times n$-incidence matrix $M_F = (m_{ij})$, having a 1 at (i, j) if (f_i, x_j) is an edge of $\mathcal{G}_F$, and a 0 otherwise. We order the entries $\{a_{ij} \mid m_{ij} = 1\}$ of Jacobian $\mathbf{J}_X F(\nu_Y, \nu_X)$, for example by lexicographic order over the pair (i, j). In this way, every $n \times n$-matrix whose pattern of nonzero coefficients is M_F identifies with a unique vector of $\mathbb{R}^K$, where

[11] Generically means: outside a set of values of Lebesgue measure zero.

K is the number of 1's in M_F. We denote by $\mathcal{J}_X F \in \mathbb{R}^K$ the image of the Jacobian $\mathbf{J}_X F(\nu_Y, \nu_X)$ obtained via this correspondence. Property 2 says:

$$\begin{array}{l}\text{There exist an open neighborhood } U \text{ of } \mathcal{J}_X F \text{ in } \mathbb{R}^K, \text{ and a subset} \\ V \subseteq U \text{ such that: } (i) \text{ the set } U \setminus V \text{ has zero Lebesgue measure, and} \\ (ii) \text{ every } \mathcal{J} \in V \text{ yields a regular matrix.}\end{array} \tag{54}$$

In the sequel, the so defined sets U and V will be denoted by

$$U_F(\nu_Y, \nu_X) \quad \text{and} \quad V_F(\nu_Y, \nu_X) \tag{55}$$

or simply U_F and V_F when no confusion can result.

Practical use of structural nonsingularity. If a square matrix is structurally singular, then it is singular. The converse is false: structural nonsingularity does not guarantee nonsingularity. Therefore, the practical use of Definition 12 is as follows: We first check if $F{=}0$ is structurally nonsingular. If not, then we abort searching for a solution. Otherwise, we then proceed to computing a solution, which may or may not succeed depending on the actual numerical regularity of the Jacobian matrix $\mathbf{J}_X F$. □

Let $F(Y, X){=}0$ be structurally nonsingular and let $\mathcal{M}$ be a complete matching for it. Using $\mathcal{M}$, graph $\mathcal{G}_F$ can be directed as follows. Edge (f_i, x_j) is directed from f_i to x_j if $(f_i, x_j) \in \mathcal{M}$, from x_j to f_i otherwise. Denote by $\vec{\mathcal{G}}_F^{\mathcal{M}}$ the resulting directed graph. The following result holds ([25], Chapter 6.10):

Lemma 14 *The strongly connected components of $\vec{\mathcal{G}}_F^{\mathcal{M}}$ are independent of $\mathcal{M}$.*

Each strongly connected component defines a *block* of F, consisting of the set of all equations that are vertices of this component. Blocks are partially ordered and we denote by $\preceq_F$ this order. Extending $\preceq_F$ to a total order (via topological sorting) yields an ordering of equations that puts the Jacobian matrix $\mathbf{J}_X F$ in *Block Triangular Form* (BTF).

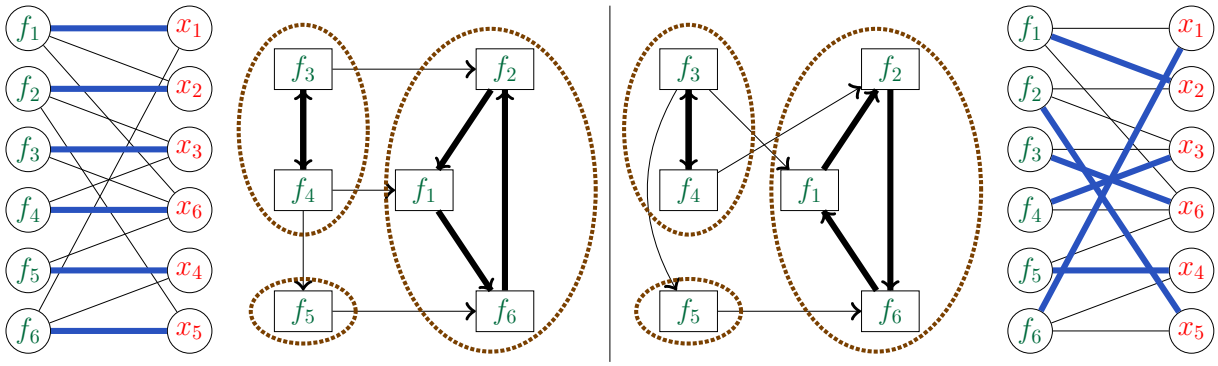


Figure 19: Illustrating Lemma 14 (we only show the projection of strongly connected components on function nodes).

This lemma is illustrated in Figure 19, inspired by lecture notes by J-Y. L'Excellent and Bora Uçar, 2010. In this figure, a same bipartite graph $\mathcal{G}$ is shown twice (left and right), with the equations sitting on the left-hand side in green, and the variables on the right-hand side in red. Two complete matchings $\mathcal{M}_1$ (left) and $\mathcal{M}_2$ (right) are shown in thick blue, with other edges of the bipartite graph being black. The restriction, to the equation vertices, of the two directed graphs $\vec{\mathcal{G}}_F^{\mathcal{M}_1}$ (left) and $\vec{\mathcal{G}}_F^{\mathcal{M}_2}$ (right) are shown on both sides. Although the two directed graphs $\vec{\mathcal{G}}_F^{\mathcal{M}_1}$ and $\vec{\mathcal{G}}_F^{\mathcal{M}_2}$ differ, the resulting block structures (encircled in dashed brown) are identical.

Nonsquare systems In our development, we will also encounter nonsquare systems of algebraic equations. For general systems (with a number of variables not equal to the number of equations, $n{\neq}m$ in (53)), call *block* a pair $\beta = (\mathbf{f}, \mathbf{x})$ collecting a subset $\mathbf{f}$ of the set of f_i's and a subset $\mathbf{x}$ of variables x_j involved in it.

Definition 15 (Dulmage-Mendelsohn) *For F=0 a general system of algebraic equations, the* Dulmage-Mendelsohn decomposition [46] *of $\mathcal{G}_F$ yields the partition of system F=0 into the following three blocks, given some matching of maximal cardinality for $\mathcal{G}_F$:*

- *block $\beta_{\mathfrak{O}}$ collects the variables and equations reachable via some alternating path from some unmatched equation;*
- *block $\beta_{\mathfrak{U}}$ collects the variables and equations reachable via some alternating path from some unmatched variable;*
- *block $\beta_{\mathfrak{E}}$ collects the remaining variables and equations.*

Blocks $\beta_{\mathfrak{O}}, \beta_{\mathfrak{U}}, \beta_{\mathfrak{E}}$ are the $\mathfrak{O}$verdetermined, $\mathfrak{U}$nderdetermined, and $\mathfrak{E}$nabled parts of F.

Statement 1 of the following lemma ensures that Definition 15 is meaningful:

Lemma 16

1. *The triple $(\beta_{\mathfrak{O}}, \beta_{\mathfrak{E}}, \beta_{\mathfrak{U}})$ defined by the Dulmage-Mendelsohn decomposition does not depend on the particular maximal matching used for its definition. We thus write $(\beta_{\mathfrak{U}}, \beta_{\mathfrak{E}}, \beta_{\mathfrak{O}}) = \mathrm{DM}(F)$.*
2. *System F is structurally nonsingular if and only if the overdetermined and underdetermined blocks $\beta_{\mathfrak{O}}$ and $\beta_{\mathfrak{U}}$ are both empty.*

A refined DM decomposition exists, which in addition refines block $\beta_{\mathfrak{E}}$ into its indecomposable block triangular form. This is a direct consequence of applying Lemma 14 to $\beta_{\mathfrak{E}}$ in order to get its BTF.

The following obvious lemma will be useful in the sequel.

Lemma 17 *Let $\mathrm{DM}(F)=(\beta_{\mathfrak{U}}, \beta_{\mathfrak{E}}, \beta_{\mathfrak{O}})$ be the Dulmage-Mendelsohn decomposition of F=0. Then, no overdetermined block exists in the Dulmage-Mendelsohn decomposition of $F \setminus \beta_{\mathfrak{O}}$.*

Comment 18 Suppose that, instead of removing from F all equations belonging to $\beta_{\mathfrak{O}}$, we only remove all unmatched equations with reference to the matching of maximal cardinality used for generating $\mathrm{DM}(F)$, and call F' the remaining subsystem of F. Then, we still get that no overdetermined block exists in the Dulmage-Mendelsohn decomposition of F' and we have $F' \supseteq (F \setminus \beta_{\mathfrak{O}})$, the inclusion being strict in general. However, this policy depends on the particular matching. In contrast, our policy is canonical since the DM decomposition is independent from the matching. Since we will rely on Lemma 17 for our compilation scheme in Section 7, adopting the finer policy would introduce a risk of nondeterminism due to the arbitrary choice of the particular complete matching selected. This is the reason for sticking with the policy stated in Lemma 17. □

6.1.3 Local versus non-local use of structural analysis

Structural analysis relies on the Implicit Function Theorem for its justification, as we have seen. Therefore, the classical use of structural analysis is local: let $F(Y, X)$=0 be a structurally nonsingular system of equations with dependent variables X, and let ν_X satisfy $F(\nu_Y, \nu_X)$=0 for given values ν_Y for Y; then, the system remains (generically) nonsingular if ν varies in a neighborhood of ν_Y. This is the situation encountered in the running of ODE or DAE solvers, since a close guess of the solution is known from the previous time step.

We are, however, also interested in invoking the structural nonsingularity of system $F(Y, X) = 0$ when no close guess is known. This is the situation encountered when handling mode changes, see our examples of Sections 2 and 3. We thus need another argument to justify the use of structural analysis in this case.

As a prerequisite we recall basic definitions regarding smooth manifolds (see, e.g., [22], Section 4.7). In the next lemma, we consider the system $F(Y, X)$=0 for a fixed value of Y, and thus we omit Y.

Lemma 19 *Let $k, m, n \in \mathbb{N}$ be such that $m < n$ and set $p = n - m$. For $\mathcal{S} \subset \mathbb{R}^n$ and $x^* = (x_1^*, \ldots, x_n^*) \in \mathcal{S}$, the following properties are equivalent:*

1. *There exists an open neighborhood V of x^* and a $\mathcal{C}^k$-diffeomorphism $F : V \to W \subset \mathbb{R}^n$ such that $F(x^*) = 0$ and $F(\mathcal{S} \cap V)$ is the intersection of W with the subspace of $\mathbb{R}^n$ defined by the m equations $w_{p+1}{=}0, \ldots, w_n{=}0$, where w_i denote the coordinates of points belonging to W.*

2. *There exists an open neighborhood V of x^*, an open neighborhood U of 0 in $\mathbb{R}^p$ and a homeomorphism $\psi : U \to \mathcal{S} \cap V$, such that ψ, seen as a map with values in $\mathbb{R}^n$, is of class $\mathcal{C}^k$ and has rank p at 0—i.e., $\psi'(0)$ has rank p.*

Statement 1 means that $\mathcal{S}$ is, in a neighborhood of x^*, the solution set of the system of equations $f_1(X){=}0, \ldots, f_m(X){=}0$ in the n-tuple of dependent variables X, where $F = (f_1, \ldots, f_m)$. Statement 2 expresses that ψ is a parameterization of $\mathcal{S}$ in a neighborhood of x^*, of class $\mathcal{C}^k$ and rank p—meaning that $\mathcal{S}$ has dimension p in a neighborhood of x^*.

In order to apply Lemma 19 to the non-local use of structural analysis, we have to study the case of square systems of equations (i.e., let $m = n$). Consider a system $F(X){=}0$, where $X{=}(x_1, \ldots, x_n)$ is the n-tuple of dependent variables and $F{=}(f_1, \ldots, f_n)$ is an n-tuple of functions $\mathbb{R}^n \to \mathbb{R}$ of class $\mathcal{C}^k$. Let $\mathcal{S} \subseteq \mathbb{R}^n$ be the solution set of this system, that we assume is non-empty. To be able to apply Lemma 19, we augment X with one extra variable z, i.e., we set $Z =_{\text{def}} X \cup \{z\}$ and we now regard our formerly square system $F(X){=}0$ as an augmented system $G(Z){=}0$ where $G(X, z) =_{\text{def}} F(X)$. Extended system G possesses n equations and $n{+}1$ dependent variables, hence, $p{=}1$, and its solution set is equal to $\mathcal{S} \times \mathbb{R}$. Let $x^* \in \mathcal{S}$ be a solution of $F = 0$. Then, we have $G(x^*, z^*){=}0$ for any $z^* \in \mathbb{R}$. Assume further that the Jacobian matrix $\mathbf{J}_X F(x)$ is nonsingular at x^*. Then, it remains nonsingular in a neighborhood of x^*. Hence, the Jacobian matrix $\mathbf{J}_X G(x, z)$ has rank n in a neighborhood V of (x^*, z^*) in $\mathbb{R}^{n+1}$. We can thus extend G by adding one more function in such a way that the resulting $(n{+}1)$-tuple $\bar{G}$ is a $\mathcal{C}^k$-diffeomorphism, from V to an open set W of $\mathbb{R}^{n+1}$. The extended system $\bar{G}{=}0$ satisfies Property 1 of Lemma 19. By Property 2 of Lemma 19, $(\mathcal{S} \times \mathbb{R}) \cap V$ has dimension 1, implying that $\mathcal{S} \cap \mathbf{proj}_{\mathbb{R}^n}(V)$ has dimension < 1. This analysis is summarized in the following lemma:

Lemma 20 *Consider the square system $F(X){=}0$, where $X{=}(x_1, \ldots, x_n)$ is the n-tuple of dependent variables and $F{=}(f_1, \ldots, f_n)$ is an n-tuple of functions $\mathbb{R}^n \to \mathbb{R}$ of class $\mathcal{C}^k, k{\geq}1$. Let $\mathcal{S} \subseteq \mathbb{R}^n$ be the solution set of this system. Assume that system $F{=}0$ has solutions and let x^* be such a solution. Assume further that the Jacobian matrix $\mathbf{J}_X F(x)$ is nonsingular at x^*. Then, there exists an open neighborhood V of x^* in $\mathbb{R}^n$, such that $\mathcal{S} \cap V$ has dimension < 1.*

The implications for the non-local use of structural analysis are as follows. Consider the square system $F(X){=}0$, where $X{=}(x_1, \ldots, x_n)$ is the n-tuple of dependent variables and $F{=}(f_1, \ldots, f_n)$ is an n-tuple of functions $\mathbb{R}^n \to \mathbb{R}$ of class $\mathcal{C}^k$. We first check the structural nonsingularity of $F{=}0$. If structural nonsingularity does not hold, then we abort solving the system. Otherwise, we proceed to solving the system and the following cases can occur:

1. the system possesses no solution;
2. its solution set is nonempty, of dimension < 1; or
3. its solution set is nonempty, of dimension ≥ 1.

Based on Lemma 20, one among cases 1 or 2 will hold, generically, whereas case 3 will hold exceptionally. Thus, structurally, we know that the system is not underdetermined. This is our justification of the non-local use of structural analysis. Note that the existence of a solution is not guaranteed. Furthermore, unlike for the local use, uniqueness is not guaranteed either.[12] Of course, subclasses of systems for which existence and uniqueness are guaranteed are of interest—the illustration examples we develop in this paper have this property.

[12] Lemma 20 only states that there cannot be two arbitrarily close solutions of $F = 0$, as its conclusion $\mathcal{S} \cap V = \{x^*\}$ involves an open neighborhood V of x^*.

6.1.4 Equations with existential quantifiers

To our knowledge, the results of this section are new. For the handling of mode change events, we will also need to develop a structural analysis for the following class of (possibly nonsquare) algebraic systems of equations with existential quantifier:

$$\exists W : F(X, W, Y)=0 \ , \tag{56}$$

where (X, W) collects the dependent variables of system $F(X, W, Y)=0$. Our aim is to find structural conditions ensuring that (56) defines a partial function $Y \to X$, meaning that the value of tuple X is uniquely defined by the satisfaction of (56), given a value for tuple Y.

At this point, a fundamental difficulty arises. Whereas (56) is well defined as an abstract relation, it cannot in general be represented by a projected system of smooth algebraic equations of the form $G(X, Y) = 0$. Such a G can be associated to (56) only in subclasses of systems.[13] In general, no extension of the Implicit Function Theorem exists for systems of the form (56); thus, one cannot apply as such the arguments developed in Section 6.1.3. Therefore, we will reformulate our requirement differently.

Say that ν_Y is *consistent* if the system of equations $F(X, W, \nu_Y)=0$ possesses a solution for (X, W). We thus consider the following property for the systen $F(X, W, Y)=0$: for every consistent ν_Y,

$$\left.\begin{array}{l} F(\nu_X^1, \nu_W^1, \nu_Y) = 0 \\ F(\nu_X^2, \nu_W^2, \nu_Y) = 0 \end{array}\right\} \implies \quad \nu_X^1 = \nu_X^2 \quad , \tag{57}$$

expressing that X is independent of W given a consistent tuple of values for Y. To find structural criteria guaranteeing (57), we will consider the algebraic system $F(X, W, Y)=0$ with X, W, Y as dependent variables (i.e., values for the entries of Y are no longer seen as being given).

Definition 21 *System* (56) *is* structurally nonsingular *if the following two conditions hold, almost everywhere when the nonzero coefficients of the Jacobian matrix* $\mathbf{J}_{X,W,Y}F$ *vary over some neighborhood:*[14] *consistent values for* Y *exist and condition* (57) *holds.*

The structural nonsingularity of (56) can be checked by using the DM decomposition of F as follows. Let $(\beta_{\mathfrak{U}}, \beta_{\mathfrak{E}}, \beta_{\mathfrak{O}}) = \mathrm{DM}(F)$ be the DM decomposition of $F(X, W, Y)=0$, with (X, W, Y) as dependent variables. We further assume that the regular block $\beta_{\mathfrak{E}}$ is expressed in its block triangular form, see Lemma 14 and comments thereafter. Let $\mathbf{B}$ be the set of all indecomposable blocks of $\beta_{\mathfrak{E}}$ and $\preceq$ be the partial order on $\mathbf{B}$ following Lemma 14. The following holds:

Lemma 22 *System* (56) *is* structurally nonsingular *if and only if:*

1. *Block* $\beta_{\mathfrak{O}}$ *is empty;*
2. *Block* $\beta_{\mathfrak{U}}$ *involves no variable belonging to* X*;*
3. *For every* $\beta \in \mathbf{B}$ *containing some variable from* X*, then,* β *contains no variable from* W*, and for every* $\beta' \prec \beta$ *and every directed edge* $(z', z) \in \vec{\mathcal{G}}_F^{\mathcal{M}}$ *such that* $z' \in \beta'$*,* $z \in \beta$*, and* $\mathcal{M}$ *is an arbitrary complete matching for* $\mathcal{G}_F$*, then* $z' \notin W$*.*

Conditions 1 and 2 speak by themselves. Regarding Condition 3, the intuition is the following. Every directed path of $\vec{\mathcal{G}}_F^{\mathcal{M}}$, originating from W and terminating in X, of minimal length, must traverse Y. Consequently, W influences X "through" Y only.

[13] Examples are linear systems for which elimination is easy, and polynomial systems for which it is doable—but expensive—using Gröbner bases.

[14] The precise meaning of this statement is the same as in Lemma 13, see the discussion therafter.

Proof *"If" part.* By Condition 1, there exist consistent values for Y. By condition 2, no variable of X belongs to the underdetermined part of $\mathrm{DM}(F)$. It remains to show that condition (57) holds. By condition (3), each indecomposable block $\beta \in \mathbf{B}$ involving a variable of X has the form $G(Z,U)=0$, where

1. Z is the n-tuple of dependent variables of $G = (g_1, \ldots, g_n)$,
2. no variable of W belongs to Z, and
3. $U \subseteq X \cup Y$ is a subset of the dependent variables of the blocks β' immediately preceding β.

Since $G=0$ is structurally regular, fixing a value for U entirely determines all the dependent variables of block β. This proves the "if" part.

"Only if" part. We prove it by contradiction. If condition 1 does not hold, then the existence of consistent values for Y is not structurally guaranteed. If condition 2 does not hold, then the variables of X that belong to $\beta_{\mathfrak{U}}$ are not determined, even for given values of W, Y. If condition 3 does not hold, then two cases can occur:

- Some $x \in X$ and $w \in W$ are involved in a same indecomposable block $\beta \in \mathbf{B}$. Then, condition (57) will not hold for x due to the structural coupling with w through block β.
- There exists an indecomposable block $\beta \in \mathbf{B}$ of the form $G(Z,U)=0$, where
  1. Z is the n-tuple of dependent variables of $G = (g_1, \ldots, g_n)$, and Z contains a variable $x \in X$,
  2. no variable of W belongs to Z, and
  3. some variable $w \in W$ belongs to U, the set of the dependent variables of the blocks β' immediately preceding β.

Hence, w influences x, structurally, which again violates (57). This finishes the proof. □

We complement Lemma 22 with the algorithm ExistQuantifEqn (Algorithm 5), which requires a system F of the form (56). If condition 1 of Lemma 22 fails to be satisfied, then $b_{\mathfrak{O}} \leftarrow \mathrm{F}$ is returned, indicating overdetermination. If conditions 2 or 3 of Lemma 22 fail to be satisfied, then $b_{\mathfrak{U}} \leftarrow \mathrm{F}$ is returned, indicating underdetermination. Otherwise, ExistQuantifEqn succeeds and returns the value T for both Booleans, together with the decomposition $F_\Sigma \cup \overline{F}_\Sigma$ of $\beta_{\mathfrak{E}}$. In this decomposition:

- Subsystem F_Σ collects the indecomposable blocks involving variables belonging to X, so that F_Σ determines X as a function of ν_Y when ν_Y is consistent;
- Subsystem $\overline{F}_\Sigma$ collects the consistency conditions, whose dependent variables belong to $W \cup Y$.

Our background on the structural analysis of algebraic equations is now complete. In the next section, we recall the background on the structural analysis of (single-mode) DAE systems.

6.2 The Σ-method for DAE systems

The DAE systems we consider are "square", i.e., have the form

$$f_j(\text{the } x_i\text{'s and their derivatives}) = 0 \tag{58}$$

where $x_1, \ldots, x_n$ denote the dependent variables and $f_1 = 0, \ldots, f_n = 0$ denote the equations. Call *leading variables* of System (58) the d_i-th derivatives[15] $x_i^{\prime d_i}$ for $i = 1, \ldots, n$, where d_i is the maximal differentiation degree of variable x_i throughout $f_1 = 0, \ldots, f_n = 0$. The problem addressed by the

[15]The notation $x^{\prime k}$ is adopted throughout this paper, instead of the more classical $x^{(k)}$, for the k-th derivative of x. See Notations 26.

Algorithm 5 ExistQuantifEqn

Require: F; **return** $(b_{\mathfrak{D}}, b_{\mathfrak{U}}, F_\Sigma, \overline{F}_\Sigma)$
1: $(\beta_{\mathfrak{U}}, \beta_{\mathfrak{E}}, \beta_{\mathfrak{D}}) = \mathrm{DM}(F)$
2: **if** condition 1 of Lemma 22 holds **then**
3: $\quad b_{\mathfrak{D}} \leftarrow \mathrm{T}$;
4: $\quad$ **if** 2 and 3 of Lemma 22 hold **then**
5: $\quad\quad b_{\mathfrak{U}} \leftarrow \mathrm{T}$;
6: $\quad\quad$ partition $\beta_{\mathfrak{E}} = F_\Sigma \cup \overline{F}_\Sigma$
7: $\quad$ **else**
8: $\quad\quad b_{\mathfrak{U}} \leftarrow \mathrm{F}$
9: **else**
10: $\quad b_{\mathfrak{D}} \leftarrow \mathrm{F}$

structural analysis of DAE systems of the form (58) is the following. Regard (58) as a system of algebraic equations with the leading variables as unknowns. If this system is structurally nonsingular, then, given a value for all the $x_i^{\prime k}$ for $i = 1, \ldots, n$ and $k = 0, \ldots, d_i - 1$, a unique value for the leading variables can be computed, structurally; hence, System (58) is like an ODE. If this is not the case, finding additional *latent equations* by differentiating suitably selected equations from (58) will bring the system to an ODE-like form, while not changing its set of solutions. Performing this is known as *index reduction.*

In our simple examples, finding latent equations was easy. In general, this is difficult and algorithms were proposed in the literature for doing it efficiently. Among them, the Pantelides algorithm [40] is the historical solution. We decided, however, to base our subsequent developments on the beautiful method proposed in [48] by J. Pryce, called the Σ-method. The Σ-method also covers the construction of the Block Triangular Form and addresses numerical issues, which we do not discuss here.

Weighted bipartite graphs We consider System (58), which is entirely characterized by its set of dependent variables X (whose generic element is denoted by x) and its set of equations $F{=}0$ (whose generic element is written $f{=}0$). We attach to (58) the bipartite graph $\mathcal{G}$ having $F \cup X$ as set of vertices and having an edge (f, x) if and only if x occurs in function f, regardless of its differentiation degree. Recall that a matching is *complete* iff it involves all equations of F.

So far, $\mathcal{G}$ is agnostic with respect to differentiations. To account for this, we further equip $\mathcal{G}$ with *weights*: to each edge $(f, x) \in \mathcal{G}$ is associated a nonnegative integer d_{fx}, equal to the maximal differentiation degree of variable x in function f. This yields a *weight* for any matching $\mathcal{M}$ of $\mathcal{G}$ by the formula $w(\mathcal{M}) = \sum_{(f,x)\in\mathcal{M}} d_{fx}$. Suppose we have a solution to the following problem:

Problem 23 *Find a complete matching $\mathcal{M}$ for $\mathcal{G}$ and integer* offsets $\{c_f \mid f \in F\}$ and $\{d_x \mid x \in X\}$, *satisfying the following conditions, where the d_{fx} are the weights as before:*

$$\begin{aligned} d_x - c_f &\geq d_{fx} \quad \textit{with equality if } (f,x)\in\mathcal{M} \\ c_f &\geq 0 \end{aligned} \tag{59}$$

Then, differentiating c_f times each function f yields a DAE system $F_\Sigma{=}0$ having the following properties. Inequality $d_x \geq c_f + d_{fx}$ holds for each $x \in X$, and $d_x = c_f + d_{fx}$ holds for the unique f such that $(f, x) \in \mathcal{M}$. Hence, the leading variables of DAE system $F_\Sigma{=}0$ are the d_x-th derivatives $x^{\prime d_x}$. Consequently, system $F_\Sigma{=}0$, now seen as a system of algebraic equations having $x^{\prime d_x}$ as dependent variables, is structurally nonsingular by Definition 12. Hence, $F_\Sigma{=}0$ is like an ODE. The integer $k = \max_{f\in F} c_f$ is called the *index* of the system.[16]

[16] We should rather say the *differentiation index* as, once again, other notions of index exist for DAEs [21] that are not relevant to our work.

Definition 24 *For F a DAE system, the solution to Problem 23 yields the DAE system $F_\Sigma=\{f'^{c_f}|f\in F\}$ together with the system of* consistency constraints *$\overline{F}_\Sigma=\{f'^{k}|f\in F, 0\leq k<c_f\}$.*

Knowing the offsets also allows transforming F into a system of index 1, by not performing the final round of differentiations. There are infinitely many solutions to Problem 23 with unknowns c_f and d_x, since, for example, adding the same $\ell\in\mathbb{N}_{\geq 0}$ to all c_f's and d_x's yields another solution. We thus seek for a *smallest* solution, elementwise. Hence, Problem 23 is the fundamental problem we must solve, and we seek a smallest solution for it.

The beautiful idea of J. Pryce is to propose a linear program encoding Problem 23. As a preliminary step, we claim that a bruteforce encoding of Problem 23 is the following: Find integers ξ_{fx}, d_x, c_f such that

$$\left.\begin{array}{r}\sum_{f:(f,x)\in\mathcal{G}}\ \xi_{fx}=1\\ \sum_{x:(f,x)\in\mathcal{G}}\ \xi_{fx}=1\\ \xi_{fx}\geq 0\end{array}\right\}\text{complete matching}$$
$$\left.\begin{array}{r}d_x-c_f-d_{fx}\geq 0\\ c_f\geq 0\end{array}\right\}\text{encodes ``}\geq\text{'' in (59)}\qquad(60)$$
$$\sum_{(f,x)\in\mathcal{G}}\ \xi_{fx}(d_x-c_f-d_{fx})=0\ \text{ encodes ``}=\text{'' in (59)}$$

We now justify our claim. Focus on the first block. Since the ξ's are integers, they can only take values in $\{0,1\}$ and one can define a bipartite graph by stating that edge (f,x) exists iff $\xi_{fx}=1$. The first two equations formalize that this graph is a matching, which is complete since all equations are involved. The second block is a direct encoding of (59) if we ignore the additional statement "with equality iff". The latter is encoded by the last constraint (since having the sum equal to zero requires that all the terms be equal to zero). Constraint problem (60) does not account for our wish for a "smallest" solution: this will be handled separately.

Following the characterization of solutions of linear programs via *complementary slackness conditions*, every solution of problem (60) is a solution of the following dual linear programs (LP), where the ξ_{fx} are real:

$$\begin{array}{rrl}primal:&\text{maximize}&\sum_{(f,x)\in\mathcal{G}}\ d_{fx}\,\xi_{fx}\\ &\text{subject to}&\sum_{f:(f,x)\in\mathcal{G}}\ \xi_{fx}=1\\ &\text{and}&\sum_{x:(f,x)\in\mathcal{G}}\ \xi_{fx}\geq 1\\ &\text{and}&\xi_{fx}\geq 0\end{array}\qquad(61)$$

$$\begin{array}{rrl}dual:&\text{minimize}&\sum_x d_x-\sum_f\ c_f\\ &\text{subject to}&d_x-c_f\geq d_{fx}\\ &\text{and}&c_f\geq 0\end{array}\qquad(62)$$

In these two problems, f ranges over F, x ranges over X, and (f,x) ranges over $\mathcal{G}$. Also, we have relaxed the integer LP to a real LP, as the two possess identical solutions in this case. Note that LP (61) encodes the search for a complete matching of maximum weight for $\mathcal{G}$. By the principle of complementary slackness in linear programming,

$$\begin{array}{l}\text{for respective optima of problems (61) and}\\ \text{(62), } \xi_{fx}>0 \text{ if and only if } d_x-c_f=d_{fx},\end{array}\qquad(63)$$

which is exactly the last constraint of (60). Using this translation into linear programs (61) and (62), it is proved in [48], Thm 3.6, that, if a solution exists to Problem 23, then a unique elementwise smallest solution exists. Based on the above analysis, the following algorithm was proposed in [48] for solving (61,62) and was proved to provide the smallest solution for (62). Let $\mathcal{G}$ be a bipartite graph with weights $\{d_{fx}|(f,x)\in\mathcal{G}\}$:

1. Solve LP (61), which gives a complete matching $\mathcal{M}$ of maximum weight for $\mathcal{G}$; any method for solving LP can be used;
2. Apply the following iteration until a fixpoint is reached (in finitely many steps), from the initial values $c_f = 0$:
   (a) $\forall x : d_x \leftarrow \max\{d_{fx} + c_f \mid (f,x) \in \mathcal{G}\}$;
   (b) $\forall f : c_f \leftarrow d_x - d_{fx}$ where $(f,x) \in \mathcal{M}$.

The reason for using the special iterative algorithm for solving the dual LP (62) is that a standard LP-solving algorithm will return an arbitrary solution, not necessarily the smallest one. The following lemma, that will be crucially put to use in Section 11.1, is an obvious consequence of the linearity of problems (61) and (62):

Lemma 25 *Let $\mathcal{G}$ be a given bipartite graph and let two families of weights $(d^1_{fx})_{(f,x)\in\mathcal{G}}$ and $(d^2_{fx})_{(f,x)\in\mathcal{G}}$ be related by $d^2_{fx} = M \times d^1_{fx}$ for every $(f,x) \in \mathcal{G}$, where M is a fixed positive integer. Then, the offsets of the corresponding Σ-method are also related in the same way: $d^2_x = M \times d^1_x$ for every variable x, and $c^2_f = M \times c^1_f$ for every function f.*

A necessary and sufficient condition for Problem 23 to have a solution is that the set of complete matchings for $\mathcal{G}$ is non-empty. We thus consider the function

$$b \leftarrow \mathsf{ExistsMatching}(\mathcal{G})\,, \tag{64}$$

which returns a Boolean b such that $b = \mathrm{T}$ iff there exists a complete matching for the bipartite graph $\mathcal{G}$ (regardless of its associated weights). In this case, step 1 of the above algorithm is guaranteed successful (since the set of complete matchings for $\mathcal{G}$ is nonempty and finite). In the sequel, we encode the above algorithm as the function

$$(\mathbf{c},\mathbf{d}) \leftarrow \mathsf{FindOffsets}(\mathcal{G},\mathbf{D})\,, \quad \text{where} \quad \left\{ \begin{array}{ll} \mathbf{c}: & F \to \mathbb{N} \\ \mathbf{d}: & X \to \mathbb{N} \\ \mathbf{D}: & F \times X \to \mathbb{N} \end{array} \right. \tag{65}$$

Having $b = \mathrm{T}$ returned by ExistsMatching() is a prerequisite to calling FindOffsets(), which is then guaranteed successful.

Algorithm 6 SigmaMethod

Require: F; **return** $(b, \overline{F}_\Sigma, F_\Sigma)$
1: $b \leftarrow \mathsf{ExistsMatching}(\mathcal{G})$.
2: **if** b **then**
3: $\quad(\mathbf{c},\mathbf{d}) \leftarrow \mathsf{FindOffsets}(\mathcal{G},\mathbf{D})$
4: $\quad(F_\Sigma, \overline{F}_\Sigma) \leftarrow \mathsf{LatentEqns}(\mathbf{c}, F)$

We encapsulate these two functions as Algorithm 6 (SigmaMethod). It requires a DAE system F=0. $\mathcal{G}$ is the weighted bipartite graph of F with weights $\mathbf{D}$. SigmaMethod returns a Boolean b and, if $b = \mathrm{T}$, a pair $(\overline{F}_\Sigma, F_\Sigma)$ defining an index-0 system F_Σ=0 and a (possibly underdetermined) system $\overline{F}_\Sigma$=0 of consistency constraints following Definition 24. In Line 4, F_Σ collects the f'^{c_f}, for $f \in F$, and $\overline{F}_\Sigma$ collects the f'^k, for $f \in F$ and $0 \le k < c_f$. This completes our background material on structural analysis.

7 Structural analysis of multimode DAE systems

In this section, we extend structural analysis, from (single-mode) DAE systems, to multimode DAE systems. As a prerequisite, we precisely define the class of multimode DAE systems considered. Since the structural analysis will actually be applied to the nonstandard semantics, we also need to define multimode *difference Algebraic Systems* (*dAE*), operating in discrete time.

7.1 Defining multimode DAE/dAE systems

In this section, the class of systems of multimode Differential/difference Algebraic Equations considered in this paper is formally defined. By analogy with DAE's, we use the acronym dAE to mean *difference* algebraic equations. Notation $x^\bullet$ denotes the shift of x, as per Definition 2.

Notations 26 Let $\mathcal{X}$ be an underlying set of variables. For $x{\in}\mathcal{X}$ and $m{\in}\mathbb{N}$, the m-differentiation and m-shift of x are denoted by x'^m and $x^{\bullet m}$, respectively. Let $\mathcal{X}'^m$ and $\mathcal{X}^{\bullet m}$ denote the set of all x'^m and $x^{\bullet m}$, for x ranging over the set $\mathcal{X}$ of variables. Let

$$\mathcal{X}^{(\prime)} =_{\text{def}} \bigcup_{m\in\mathbb{N}} \mathcal{X}'^m \text{ and } \mathcal{X}^{(\bullet)} =_{\text{def}} \bigcup_{m\in\mathbb{N}} \mathcal{X}^{\bullet m} \ . \tag{66}$$

For $X{\subset}\mathcal{X}$, sets $X'^m, X^{\bullet m}, X^{(\prime)}, X^{(\bullet)}$ are defined in a similar way. □

The following assumption will be in force in the remainder of this paper:

Assumption 1 *All the functions and predicates over $\mathcal{X}^{(\prime)}$ are assumed to involve only finitely many of the x'^m for $x{\in}\mathcal{X}$ and $m{\in}\mathbb{N}$. The same holds regarding $\mathcal{X}^{(\bullet)}$.*

7.1.1 Syntax and meaning

From the discussion in Section 3.6 regarding the Cup-and-Ball example, we know we need to type the modes as either *long* or *transient*, since the two situations require different index reduction techniques. This justifies the consideration of two different kinds of guarded equations.

Definition 27 (mDAE/mdAE syntax) Multimode DAE systems (mDAEs), *respectively* multimode dAE systems (mdAEs), *are defined by the following syntax:*

$$\begin{aligned} S &::= s \mid S; s \\ s &::= g \mid e \\ g &::= \gamma = b && (67)\\ e &::= \texttt{if}\ \gamma\ \texttt{then}\ f{=}0 && (68)\\ &\mid \texttt{when}\ \gamma\ \texttt{then}\ f{=}0 && (69) \end{aligned}$$

mDAE S is a finite set of s; s is either a *guard evaluation* g (67), or a *guarded equation* e, of two different kinds (68) or (69); f is a smooth scalar function over $\mathcal{X}^{(\prime)}$, and b is a Boolean expression of predicates over $\mathcal{X}^{(\prime)}$. The same holds for the syntax of mdAE, with $\mathcal{X}^{(\bullet)}$ replacing $\mathcal{X}^{(\prime)}$.

Notations 28 Let e be a guarded equation. Its *body* $f{=}0$ and *guard* γ are denoted by $f(e)$ and $\gamma(e)$. For S an mDAE or mdAE system, we denote by $X(S)$, $E(S)$, and $\Gamma(S)$ its sets of numerical variables, guarded equations, and guards, respectively. The mention of S is omitted when it is clear from the context. Finally, for e : `if/when` γ `then` $f{=}0$ a guarded equation following Definition 27, we define

$$e^\bullet/e' \ : \ \texttt{if/when}\ \gamma\ \texttt{then}\ f^\bullet/f'{=}0 \tag{70}$$

where $f^\bullet$ follows Definition 2, and, for E a set of equations, Notations 26 are adapted so that $E'^m, E^{\bullet m}, E^{(\prime)}, E^{(\bullet)}$ can be considered. We insist that, in (70), the guard γ is left unchanged. □

Meaning The meanings of `if`-equation (68) and `when`-equation (69) differ:

- In the `if`-equation (68), equation $f{=}0$ is enabled if and only if the guard γ holds. Otherwise, this equation is disabled.
- In the `when`-equation (69), equation $f{=}0$ is enabled exactly at the events when guard γ switches from F to T. Otherwise, this equation is disabled. □

For mdAE systems, the formalization of the second statement is straightforward: equation $f{=}0$ is enabled when

$$\gamma^{\uparrow} \text{ holds, where } \gamma^{\uparrow} =_{\text{def}} (\texttt{not } {}^{\bullet}\gamma) \texttt{ and } \gamma \ . \tag{71}$$

For mDAE systems: equation $f{=}0$ is enabled when

$$\gamma^{\uparrow} \text{ holds, where } \gamma^{\uparrow} =_{\text{def}} (\texttt{not } \gamma^{-}) \texttt{ and } \gamma \tag{72}$$

where γ^{-} denotes the left-limsup of γ at the current instant, defined by:

$$\begin{array}{rcl} \gamma^{-}(t) & =_{\text{def}} & \limsup_{s \nearrow t} \gamma(s) \\ & = & \texttt{if } \exists s_n \nearrow t : \gamma(s_n){=}\text{T} \texttt{ then T else F} \end{array}$$

The left-limsup is always defined and coincides with the left limit when the latter exists.

Comment 29 Why bother with the two types of equations, since `when`-equations are mapped to `if`-equations by mapping γ to $\gamma^{\uparrow}$? The motivation was given in Section 3.6. We need to type the modes as either *long* or *transient*, since the two situations require different index reduction techniques. This is precisely what our `when`-equations guarantee, since, by construction, switching from F to T occurs in a single nonstandard instant. *Hence, by construction,* `when`*-equations are active at transient modes only.* Actually, this distinction between `if`-equations and `when`-equations already exists in object-oriented DAE-based languages such as Modelica. □

Regarding `if`-equations, we state the following assumption:

Assumption 2 *In* `if`*-equations, guard γ holds true for a (standard) positive duration.*

It is the responsibility of the programmer to make sure that `if`-equations are properly used so that Assumption 2 holds.

7.1.2 Long versus transient modes

For our theoretical development and the presentation of algorithms, it will be convenient to simplify our syntax by replacing `when`-equations by `if`-equations using (71) and Assumption 2. Our syntax of Definition 27 with arbitrary guards is thus replaced with the following (equivalent) syntax:

Definition 30 (mDAE/mdAE revisited) Multimode DAE systems (mDAEs), *respectively* multimode dAE systems (mdAEs), *are defined by the following syntax:*

$$\begin{array}{rcl} S & ::= & s \mid S; s \\ s & ::= & g \mid e \\ g & ::= & \gamma = b \\ e & ::= & \texttt{if } \gamma \texttt{ then } f{=}0 \end{array} \tag{73}$$

where b is a `long`/`transient`-typed *Boolean expression of predicates over $\mathcal{X}^{(\prime)}$ (respectively $\mathcal{X}^{(\bullet)}$).*

From now on, and unless otherwise specified, we will use the modified syntax of Definition 30. The notion of mode is then easily introduced.

Definition 31 (mode) *For S a mDAE or mdAE, a* mode *is a valuation, in $\{\text{F}, \text{T}\}$, of all of its guards $\gamma \in \Gamma(S)$. The mode is called* transient *if at least one guard typed* `transient` *takes the value* T *in it; otherwise, it is called* long. *Modes will be generically denoted by the symbol μ.*

A mode enables a subset of the equations $f=0$ and disables the other ones.

Nonstandard semantics. An mDAE S is transformed to a (nonstandard) mdAE ${}^\star S$ through the following syntactic transformation:

$$ {}^\star S \quad =_{\text{def}} \quad S\left[x' \text{ is replaced with } \frac{x^\bullet - x}{\partial}\right] \tag{74}$$

Since we perform the structural analysis on the nonstandard semantics, we must develop a structural analysis for mdAE systems, operating in discrete time. Therefore, in the rest of this section, we work in nonstandard semantics. □

Comment 32 (Progressive evaluation of the modes) It may be that some guards have their value unknown when the execution of the current instant starts; this was, for example, the case with the Cup-and-Ball example in its first form given in Section 1.2.2. We can use the known enabled equations to evaluate part of the variables, hoping that some of the yet unknown guards get their value defined. This yields more equations getting enabled/disabled, allowing for more guards being evaluated. If this process leads to the evaluation of all guards, then our execution scheme succeeds; otherwise, we report a failure due to a logico-numerical fixpoint equation, as was the case in Section 1.2.2.

This progressive evaluation of guards becomes problematic when both long and transient modes are considered. Indeed, different structural analyses are used for each case (see Section 3.3 for the long mode case of the Cup-and-Ball example, versus Section 3.5 for the transient mode case). Unfortunately, we do not know which one should be used if the mode is only partially known, since the long/transient qualification is only known when all guards have been evaluated. Consequently, we have to choose between the following two incompatible features for our compilation method:

1. either we support the progressive evaluation of modes, but are unable to handle transient modes;
2. or we support both long and transient modes, but we do not support the progressive evaluation of modes.

Since many physically meaningful models involve transient modes, our opinion is that the second option is much more useful in practice, hence it is adopted in our work. □

Comment 6, regarding the Cup-and-Ball example, discussed what we miss by favoring the second option. Indeed, delaying the effect of guards was the appropriate correction to remove logico-numerical fixpoint problems in the Cup-and-Ball example. Remember that the so introduced delay is infinitesimal, so this does not really matter.[17]

Therefore, throughout the rest of this work, Definition 30 is modified to restrict ourselves *by the syntax* to systems in which all guards can be evaluated at the initialization of each nonstandard instant:[18]

Definition 33 (mdAE modified) Multimode dAE systems (mdAEs), *are defined by the following syntax:*

$$\begin{aligned} S &::= s \mid S; s \\ s &::= g \mid e \\ g &::= \gamma^\bullet = b \\ e &::= \texttt{if } \gamma \texttt{ then } f=0 \end{aligned} \tag{75}$$

where b is a `long`/`transient`-typed *Boolean expression of predicates over $\mathcal{X}^{(\bullet)}$.*

Notations 34 Referring to (75), we write $b(\gamma)$ to refer to the expression defining $\gamma^\bullet$. □

From now on, by mdAE, we will mean a system of the form given in Definition 33.

[17] This is unlike Synchonous Languages for discrete time systems, in which the restriction resulting from adopting Definition 33 would be significant.

[18] With this new syntax, the original forms for the Cup-and-Ball and Westinghouse examples (given in the introduction) could not be expressed.

7.2 Structural analysis: intuition

In this section, we give an intuition about how to extend the structural analysis from dAE systems to mdAE systems. To simplify our explanation, *we consider the restricted case in which both modes (previous and current) are long.* Also, it will be convenient to assume, for the two modes before and after the change, that we have two index-1 models, of the form

$$S_i : \left\{ \begin{array}{rcl} X_i' & = & F_i(X_i, Y_i) \\ 0 & = & H_i(X_i) \end{array} \right. , \text{ for } i = 0, 1 ,$$

where indices 0 and 1 refer to the system before and after the change. Observe that system state X_i may collect different variables before and after change, and so does the algebraic tuple Y_i. Of course, the dynamics F_i, H_i are subject to changes as well. Let $\mathbf{J}_i =_{\text{def}} \mathbf{J}_{X_i} H_i$ denote the Jacobian of H_i w.r.t. X_i. Hence, the system augmented with its latent equations

$$S_{i\Sigma} : \left\{ \begin{array}{rcl} X_i' & = & F_i(X_i, Y_i) \\ 0 & = & H_i(X_i) \\ 0 & = & \mathbf{J}_i(X_i) X_i' \end{array} \right. , \text{ for } i = 0, 1 ,$$

seen as a system of algebraic equations with X_i', Y_i as dependent variables, is structurally nonsingular. Equivalently, we may replace, in this system, the differentiation of the algebraic constraint, $\frac{d}{dt} H_i(X_i)$, by its shifting $H_i(X_i^\bullet)$ (their bipartite graphs are equal):

$$S_{i\Sigma} : \left\{ \begin{array}{rcl} X_i' & = & F_i(X_i, Y_i) \\ 0 & = & H_i(X_i) \\ 0 & = & H_i(X_i^\bullet) \end{array} \right. , \text{ for } i = 0, 1 . \tag{76}$$

Form (76) can be seen as the result of applying Pantelides' or Pryce's structural analysis in each mode, before and after change.

Comment 35 (key idea) *Our key idea is to handle the mode change as a task of reconciling the possible conflict between the predictions generated by the previous mode and the consistency conditions associated to the new mode.* Indeed, at the instant of mode change, there is a possible conflict between:

- on the one hand, the value for X_0 predicted by model $S_{0\Sigma}$ which sets the value of the derivative X_0' just before the change;
- on the other hand, the consistency constraints on the pair X_1, Y_1 resulting from model $S_{1\Sigma}$ after the change. □

To perform this reconciliation, we move to the nonstandard semantics by *syntactically replacing*, in (76), the derivatives x' by their first-order Euler approximations with infinitesimal time step ∂:

$$x'(t) \underbrace{\leftarrow}_{\text{is replaced by}} \frac{x(t+\partial) - x(t)}{\partial} = \frac{x^\bullet(t) - x(t)}{\partial} .$$

In addition, we replace, in (76), the derivative $\frac{d}{dt} H_i(X_i) = \mathbf{J}_i(X_i) X_i'$ by the time shift $H_i(X_i^\bullet)$. This substitution is legitimate, since the bipartite graph of the system is left unchanged by performing this replacement.

Performing the above substitutions in (76) and sampling it on the nonstandard discrete time basis $\mathbb{T}=\{0, \partial, 2\partial, \dots\}$ yields a discrete-time two-modes system. At the instant of mode change, we inherit the following system of algebraic equations:

$$S = \underbrace{{}^\bullet S_{0\Sigma}}_{\text{previous instant}} \bigcup \underbrace{S_{1\Sigma}}_{\text{current instant}} \tag{77}$$

where

$$ {}^{\bullet}S_{0\Sigma}: \left\{ \begin{array}{rcl} \frac{X_0 - {}^{\bullet}X_0}{\partial} & = & F_0({}^{\bullet}X_0, {}^{\bullet}Y_0) \\ 0 & = & H_0({}^{\bullet}X_0) \\ 0 & = & H_0(X_0) \end{array} \right. , \; S_{1\Sigma}: \left\{ \begin{array}{rcll} \frac{X_1^{\bullet} - X_1}{\partial} & = & F_1(X_1, Y_1) & (e_0) \\ 0 & = & H_1(X_1) & (e_1) \\ 0 & = & H_1(X_1^{\bullet}) & (e_1^{\bullet}) \end{array} \right. \tag{78} $$

and $\cup$ refers to the union of sets of equations. For each state variable x that is shared between the two systems, i.e., $x \in X_0 \cap X_1$, the value for x must be identical in ${}^{\bullet}S_{0\Sigma}$ and $S_{1\Sigma}$. To avoid duplicates of a same equation while taking the union of previous and current systems in S, *we identify previous and current equations that are identical in their syntax, if any.* The dependent variables of S are the X_0 inherited from $S_{0\Sigma}$ and the $X_1, Y_1, X_1^{\bullet}$ inherited from $S_{1\Sigma}$. When seen as sets of variables, X_0 on the one hand, and $X_1, Y_1, X_1^{\bullet}$ on the other hand, possess in general a nonempty intersection. Being the result of a structural analysis, system $S_{1\Sigma}$ is:

- structurally nonsingular, when seen as a system of algebraic equations in its leading variables $Y_1, X_1^{\bullet}$, for given values for X_1; and
- possibly structurally underdetermined, when seen as a system of algebraic equations in all its variables $X_1, Y_1, X_1^{\bullet}$ (ensuring consistent initialization).

Focus on $S_{1\Sigma}$ in decomposition (77). Denote by Φ the subsystem of $S_{1\Sigma}$ collecting all its equations that also belong to ${}^{\bullet}S_{0\Sigma}$, and are thus already solved.

The remaining system $S_{1\Sigma} \setminus \Phi$ must then be solved for its dependent variables, which consist of the subset of $X_1 \cup Y_1 \cup X_1^{\bullet}$ collecting all variables not yet determined by ${}^{\bullet}S_{0\Sigma}$.

To prepare for this, we submit $S_{1\Sigma} \setminus \Phi$ to the Dulmage-Mendelsohn (DM) decomposition, see Definition 15: $(\beta_{\mathfrak{U}}, \beta_{\mathfrak{E}}, \beta_{\mathfrak{O}}) \leftarrow \mathrm{DM}(S_{1\Sigma} \setminus \Phi)$. If $\beta_{\mathfrak{U}} = \beta_{\mathfrak{O}} = \emptyset$, then $S_{1\Sigma} \setminus \Phi$ is structurally nonsingular by Lemma 16 and we solve it. Otherwise, the following can occur:

Case $\beta_{\mathfrak{U}} \neq \emptyset$. This points to an underspecified subsystem at the instant of mode change. This information is returned to the designer for a correcting action.

Case $\beta_{\mathfrak{U}} = \emptyset$ *and* $\beta_{\mathfrak{O}} \neq \emptyset$. This points to a conflict between ${}^{\bullet}S_{0\Sigma}$ and $S_{1\Sigma}$ and is handled differently. We solve this conflict by applying Principle 1 of causality. Accordingly, the equations belonging to $\beta_{\mathfrak{O}}$ are removed from the consistency equations (e_1) of the new system $S_{1\Sigma}$. We thus keep the system $S_{1\Sigma}^{\downarrow} =_{\mathrm{def}} \{(e_0), (e_1) \setminus \beta_{\mathfrak{O}}, (e_1^{\bullet})\}$. By Lemma 17, since the system $\{(e_0), (e_1^{\bullet})\}$ is structurally nonsingular, performing the DM decomposition on system $S_{1\Sigma}^{\downarrow}$ returns a triple with an empty overdetermined block. Hence, the so obtained reduced system $S_{1\Sigma}^{\downarrow}$ is structurally regular and ready to be solved.

Comment 36 If the two systems $S_{i\Sigma}, i = 0, 1$ are identical (there is no mode change), then $X_1 = X_0, Y_1 = Y_0, F_1 = F_0$, and $H_1 = H_0$, so that Φ consists of the consistency subsystem $0 = H_1(X_1)$, see the definition of $S_{1\Sigma}$ in (78). Hence, $S_{1\Sigma} \setminus \Phi$ coincides in this case with the structurally nonsingular system obtained by performing index reduction (see Definition 24), and we are left with the usual structural analysis for DAE.

The reconciliation may take several nonstandard instants following the mode change, where less and less equations from $S_{1\Sigma}$ are erased when applying DM, until we reach the steady regime. The interpretation of the erasure of some equations from $S_{1\Sigma}$ is that their satisfaction gets postponed for a few instants. Recall that we are in nonstandard semantics, so that all of this will occur in a total zero duration, in standard real time.

In the rest of this section, we formalize this intuition and generalize it.

7.3 Structural Analysis of long modes

In this section we develop the structural analysis of mdAE systems for the subclass of systems possessing only long modes. Within each long mode, we simply reuse the existing structural analysis

based on the Σ-method. The only difficulty sits in the mode changes, for which we use conflict reconciliation as explained above. Throughout this section, we work in the nonstandard semantics as given by Equation (74).

7.3.1 Constructive Semantics

We reuse the techniques of *constructive semantics,* first introduced in the context of reactive synchronous programming languages [12, 9, 10, 47], for the purpose of grounding compilation on solid mathematical foundations. For synchronous languages, the execution of a program proceeds through successive *reactions*, by which discrete time progresses from the current instant to the next one. The compilation of a program consists in generating the code for executing a reaction. This task is formalized through the notion of *constructive semantics*, which relies on the following two pillars:

1. The specification of the set of *atomic actions*, which are effective, non-interruptible, state transformations. For synchronous languages, atomic actions consist of: (*a*) the evaluation of a single expression, and (*b*) control flow actions. Executing an instance of an atomic action is referred to as performing a *micro-step*.

2. The *causality analysis* of the reaction, which is a partial order on the set of micro-steps, abstracting the dependencies between them. A scheduling of the micro-steps is correct if and only if it complies with the causality analysis.

The *constructive semantics* consists in decomposing a reaction as the symbolic execution of a sequence of micro-steps subject to causality constraints. Intermediate stages of this execution are represented by the 'status' of each variable, belonging to the abstract alphabet {not evaluated/evaluated}. The assignment of a status to all variables is called a *configuration*. Such executions are monotonic, in that the so produced sequence of configurations is increasing with respect to the product order derived from the order *not evaluated* < *evaluated* for each variable. The constructive semantics is complete if it terminates by having all variables evaluated. Failure occurs if an incomplete sequence cannot be extended.

This approach is suited to mdAE systems, since it will provide the formal support for correctly chaining the different actions composing the structural analysis, and describing the resulting increase in knowledge on all involved equations, variables and guards. For mdAE systems, the *atomic actions*, however, consist of:

(*a*) the evaluation of a guard;

(*b*) manipulations of systems of equations: adding latent equations or deleting conflicting equations at mode changes; and

(*c*) solving a block of algebraic equations.

Actions (*a*) and (*b*) participate in the structural analysis, whereas (*c*) is the runtime solving of systems of equations to evaluate the leading variables. To keep this evaluation symbolic, we abstract away the values of numerical variables and regard the values of guards as oracles, so that all the possible valuations must be explored.

Statuses and Contexts The different statuses of the guards, variables and equations of an mdAE are the following: an evaluated guard may be *true* or *false*; a variable may be *not evaluated*, or *evaluated*; an equation may be *not evaluated*, *disabled* (its guard is false), or *solved*. This is summed up in Table 1. There is no need to consider the status *not evaluated* for a guard, since guards can be evaluated at the initialization of each nonstandard instant, by Definition 33.

Unlike for single-mode DAEs, the set of equations defining the dynamics of an mdAE is mode-dependent. This information should therefore be traced as part of the constructive semantics of mdAE systems. To capture this, we tag as *irrelevant* in the considered mode all the variables,

	I	U	F	T
guard	irrelevant		evaluated to F	evaluated to T
variable	irrelevant	not evaluated		evaluated
equation	irrelevant	not evaluated	disabled	solved

Table 1: The partially ordered domain $D = \{\text{I}, \text{U}, \text{F}, \text{T}\}$ and its interpretation for guards, variables, and equations.

guards, and equations that are not used to define the dynamics of the mdAE system in this mode. Formally, the domain of statuses is the partially ordered set $(D, <)$ defined by

$$D = \{\text{I}, \text{U}, \text{F}, \text{T}\} \quad \text{and} \quad \text{I} < \text{U} < \text{F}, \text{T} \ . \tag{79}$$

The interpretation of this domain for variables, equations and guards, is summarized in Table 1 and detailed next:

- The minimal element I is used to represent the fact that a variable, a guard, or an equation is *irrelevant*.
- Value U means that a variable or equation has *not been evaluated yet*. At the beginning of a time step, only state variables are known, and all other variables are set to U, reflecting that their numerical values are not known yet.
- Maximal element T has different meanings, depending on whether it applies to a variable, a guard or an equation. In the case of a variable, it means that the numerical value of the variable has been computed, whatever it could be. For a guard, it means that the guard has been evaluated to true. For an equation, it means that the equation has been solved.
- Maximal element F also has different meanings, depending on whether it applies to a guard or an equation. In the context of a guard, it means that the guard has been evaluated to false. When it applies to an equation, it means that the equation is disabled, i.e., its guard is false. This value does not apply to variables.

We call *status* the assignment, to each guard, variable, and equation, of a value over D. It is formally defined as follows:

Definition 37 (status) *Let S be an mdAE system. Its set $\mathbb{V}$ of* S-variables[19] *is*

$$\mathbb{V} \quad =_{\text{def}} \quad \Gamma \cup X^{(\bullet)} \cup E^{(\bullet)} \tag{80}$$

(using Notations 26 and 28). A status *σ of S is a valuation in D of the S-variables, that is, a mapping $\sigma : \mathbb{V} \to D$. A status is called* finite *if it is equal to* I*, except for a finite number of S-variables. The set of statuses is partially ordered by the product order: $\sigma_1 \leq \sigma_2$ if and only if for all $v \in \mathbb{V}$, $\sigma_1(v) \leq \sigma_2(v)$.*

We denote by $\text{I} \nearrow \text{U}$, $\text{U} \nearrow \text{F}$, and $\text{U} \nearrow \text{T}$ the increasing changes of status for a given S-variable.

Enabled Equations and Leading Variables. Let σ be a status. Equation $e^{\bullet m}$ is *enabled in σ* (respectively *disabled in σ*) if and only if $\sigma(\gamma) = \text{T}$ (respectively $\sigma(\gamma) = \text{F}$), where γ is the guard of $e^{\bullet m}$, see (70). Denote by

$$\mathit{Enab}(\sigma) \text{ (respectively } \mathit{Disab}(\sigma))$$

the set of equations that are enabled (respectively disabled) in σ. Recall that, for any finite status σ, these sets are finite. The set $\mathit{Enab}(\sigma)$ defines the DAE system that is enabled at status σ. The

[19] The prefix "**S**" is reminiscent of constructive **S**emantics.

leading variables in status σ are defined, with reference to this enabled DAE system, following the introduction of Section 6.2. We define the set

$$Undef(\sigma) \quad =_{\text{def}} \quad \{v \in \mathbb{V} \mid \sigma(v) \leq \text{U}\}$$

collecting the S-variables that are either irrelevant or not evaluated in status σ. We are now ready to introduce micro-steps, and runs as finite sequences of micro-steps.

Definition 38 (micro-step) *A* micro-step *is a transformation, via some atomic action, of a status σ to a status σ' by updating the values of a finite subset of S-variables, from* I *to* U*, or from* U *to* T *or* F*.*

A run is a finite sequence of micro-steps. It is complete when every equation is either irrelevant, disabled, or solved, and all the enabled leading variables are evaluated. Formally (the reader is referred to Definition 37 for the notions used):

Definition 39 (runs) *Given a finite initial status σ_0, a* run *of mdAE system S is a finite increasing sequence of statuses*

$$\sigma_0 < \sigma_1 < \cdots < \sigma_k < \sigma_{k+1} < \cdots < \sigma_{\mathsf{end}} \tag{81}$$

such that, for every $k < \mathsf{end}$, σ_{k+1} is obtained from σ_k by performing a micro-step. The final status σ_{end} is called complete *if, in σ_{end}, all equations e_i have either the value* T *or* F *and no leading variable has the value* U*. A run $\sigma_0 < \ldots < \sigma_{\mathsf{end}}$ of mdAE system S is* complete *if and only if its final status σ_{end} is complete.*

Handling conflicts between past and present. To handle the conflicts between the previous and current instants when a mode change occurs, we will need to keep track of the possible sources of conflict between previous and current instants. If equation $e^{\bullet(m+1)}, m{\geq}0$, was solved at the previous instant (meaning that $\sigma(e^{\bullet(m+1)}) = \text{T}$ for the final status of that instant), then $e^{\bullet m}$ is a possible source of conflict at the current instant. The set of all such equations is called the *context* at the current instant. All the variables involved in the context are already evaluated at the initial status of the current run (value T).

If, however, equation $e^{\bullet m} \in \Delta$ is also enabled at status σ of the current instant, then $e^{\bullet m}$ is no longer a source of conflict, but rather a satisfied consistency equation. We call it a *fact* at σ. Formally:

Definition 40 (contexts and facts) *Let S be an mdAE system. Its* context Δ *at the current instant, and* set of facts $\Phi(\sigma)$ *at the current status, are defined as follows:*

$$\Delta = \left\{e^{\bullet m} \mid \text{the final status of } e^{\bullet(m+1)} \text{ at previous instant is T}\right\}$$
$$\Phi(\sigma) = \Delta \cap Enab(\sigma)$$

By abuse of notation, we will also denote by Δ and Φ the sets of functions $f(e)$, for e ranging over Δ and Φ.

To prepare for the algorithm computing a run for given initial status σ_0 and context Δ, we now collect a few useful auxiliary algorithms.

7.3.2 Auxiliary algorithms

Tick algorithm ***(Algorithm 7)*** When a run is complete, the system can proceed to the next time step by executing the algorithm Tick. Tick requires a complete (final) status σ and returns a pair $(\sigma^\circ, \Delta^\circ)$ consisting of an initial status σ° and a context Δ°, for use as initial conditions for the next run.

Performing Tick has the effect of shifting backward defined variables, and setting all other S-variables $v \in \mathbb{V}$ to either U, if v is relevant to the mdAE S, or I (irrelevant) otherwise. Guards are set either to I or to the value $\sigma(b(\gamma)) \in \{\text{F}, \text{T}\}$, using Notations 34 and the definition (75) for $\gamma^\bullet$. Thus, the value of all guards at the next instant is set. The new context is defined to be the set of equations that are known to be satisfied in the next instant. Tick is not increasing and it does not have to be, since it applies when moving to the next time step.

Algorithm 7 Tick

Require: σ, a complete status; **return** pair $(\sigma^\circ, \Delta^\circ)$.

1: $\mathsf{Tick}(\sigma) =_{\text{def}} (\sigma^\circ, \Delta^\circ)$, where:

$$\begin{aligned}
\sigma^\circ(\gamma) &= \textbf{if } \gamma \text{ is involved in } S \textbf{ then } \sigma(b(\gamma)) \textbf{ else } \textsc{i} \\
\sigma^\circ\big(x^{\bullet m}\big) &= \textbf{if } \sigma\big(x^{\bullet m+1}\big) = \textsc{t} \textbf{ then } \textsc{t} \textbf{ else} \\
&\quad\ \textbf{if } x^{\bullet m} \text{ is a variable of } S \textbf{ then } \textsc{u} \textbf{ else } \textsc{i} \\
\sigma^\circ\big(e^{\bullet m}\big) &= \textbf{if } e^{\bullet m} \text{ is a variable of } S \textbf{ then } \textsc{u} \textbf{ else } \textsc{i} \\
\Delta^\circ &= \big\{ e^{\bullet m} \,\big|\, e{\in}E, m{\in}\mathbb{N} : \sigma\big(e^{\bullet m+1}\big){=}\textsc{t} \big\}
\end{aligned}$$

IndexReduc **algorithm** ***(Algorithm 8)*** Given a system G with its set of leading variables determined by the status σ, the IndexReduc algorithm is a renaming of the Σ-method (Algorithm 6), together with a refined interpretation of its result. *It is understood that we use the discrete-time adaptation of it, in which forward shift replaces differentiation but everything else remains unchanged.*

Algorithm 8 IndexReduc

Require: (G, σ); **return** $(\sigma, b, G_\Sigma, \overline{G}_\Sigma)$

1: $(b, G_\Sigma, \overline{G}_\Sigma) \leftarrow \mathsf{SigmaMethod}(G)$ with dependent variables set by σ; increase σ

2: **if** b **then return** $(G_\Sigma, \overline{G}_\Sigma)$

Recall that this algorithm requires a square system $G{=}0$ and returns a success/failure flag b, as well as, in case of success, a pair consisting of $G_\Sigma{=}0$, a structurally nonsingular system determining some leading variables, and the system $\overline{G}_\Sigma{=}0$ collecting the consistency equations. As a consequence, the status of every latent equation added by the Σ-method is updated as $\textsc{i} \nearrow \textsc{u}$.

SolveConflict **algorithm** ***(Algorithm 9)*** This algorithm requires a system $K{=}0$ and a status σ for all variables involved in K. System K is submitted to the DM decomposition (Line 1); in doing so, the dependent variables are those with status $\textsc{u}$. Possible conflicts are collected in $\beta_\mathfrak{O}$. If the latter is nonempty, corresponding equations are deleted (Line 3). The status is updated as $\textsc{u} \nearrow \textsc{f}$ for every removed equation. The so reduced system is again submitted to the DM decomposition (Line 4), which, by Lemma 17, will return an empty overdetermined block: conflicts are removed. If block $\beta_\mathfrak{U}$ is nonempty, then system K is underdetermined. Therefore, $b = \textsc{f}$ is returned (Line 5). Otherwise, $b = \textsc{t}$ and $H = \beta_\mathfrak{E}$ are returned (Line 6).

Algorithm 9 SolveConflict

Require: (K, σ); **return** (σ, b, H)

1: $(\beta_\mathfrak{U}, \beta_\mathfrak{E}, \beta_\mathfrak{O}) \leftarrow \mathrm{DM}(K)$ with dependent variables set by σ

2: **if** $\beta_\mathfrak{O} \neq \emptyset$ **then**

3: $\quad K \leftarrow K \setminus \beta_\mathfrak{O}$; increase σ

4: $\quad (\beta_\mathfrak{U}, \beta_\mathfrak{E}, \emptyset) \leftarrow \mathrm{DM}(K)$ with dependent variables set by σ

5: $b \leftarrow [\beta_\mathfrak{U} = \emptyset]$

6: **if** b **then** $H \leftarrow \beta_\mathfrak{E}$

Eval **algorithm**

$$\sigma \leftarrow \mathsf{Eval}(H, \sigma) \tag{82}$$

This algorithm requires a status σ and a structurally regular system H, enabled at σ. It evaluates H for its dependent variables, which sets the values of the leading variables. The corresponding change in status is U $\nearrow$ T for both the variables that are evaluated and the equations that are solved.

7.3.3 Executing a nonstandard instant

Algorithm 10 ExecRun

Require: (σ, Δ); **return** $(\mathit{Fail}, (\sigma^\circ, \Delta^\circ))$
1: $\Phi \leftarrow \{e \in \Delta \mid \sigma(\gamma(e)) = \mathrm{T}\}$
2: $G \leftarrow [\mathit{Enab}(\sigma) \cap \mathit{Undef}(\sigma)]$
3: $(\sigma, b, G_\Sigma, \overline{G}_\Sigma) \leftarrow \mathsf{IndexReduc}(G, \sigma)$; increase σ
4: **if** $\neg b$ **then return** $\mathit{Fail}(\sigma)$
5: **else**
6: $\quad(\sigma, b, H) \leftarrow \mathsf{SolveConflict}((G_\Sigma \cup \overline{G}_\Sigma) \setminus \Phi, \sigma)$; increase σ
7: $\quad$**if** $\neg b$ **then return** $\mathit{Fail}(\sigma)$
8: $\quad$**else**
9: $\quad\quad\sigma \leftarrow \mathsf{Eval}(\sigma, H)$; increase σ
10: $\quad\quad(\sigma^\circ, \Delta^\circ) \leftarrow \mathsf{Tick}(\sigma)$

ExecRun requires a finite status σ and a finite context Δ to serve as initial conditions for the run. It returns, either a documented *Fail*, or a final value for status σ, as well as values for σ° and Δ° to serve as initial conditions for the next instant. A line-by-line description of ExecRun is given next.

Line 1 Not all the equations belonging to the context are active in the current instant. The active ones are collected in the set Φ of *facts*.

Line 2 System G is set to the enabled guarded equations that are not evaluated yet in status σ.

Line 3 We submit G to index reduction (IndexReduc algorithm, Algorithm 8).

Line 4 If IndexReduc returns b=F, then ExecRun returns $\mathit{Fail}(\sigma)$ and stops.

Line 5 Otherwise, ExecRun returns a pair consisting of G_Σ=0, a structurally nonsingular system determining some leading variables, and the system $\overline{G}_\Sigma$=0 collecting consistency equations. In this case, the algorithm can further progress.

Line 6 Taking the set Φ of facts into account, the possible conflict is solved using the SolveConflict algorithm (Algorithm 9). When applying SolveConflict, the dependent variables of $(G_\Sigma \cup \overline{G}_\Sigma) \setminus \Phi$ include all the variables (both leading and non-leading) of the current system that have status $\sigma = \mathrm{U}$, i.e., whose value was not set by executing the previous nonstandard instant.

Line 7 If SolveConflict fails, ExecRun returns $\mathit{Fail}(\sigma)$ and stops.

Line 9 Otherwise, SolveConflict returns a structurally regular system H, which is solved using Eval, and the so reached status σ is complete.

Line 10 Since σ is complete, Tick (Algorithm 7) is performed, so that the values of guards, the status of all S-variables and the context are known for the next nonstandard instant.

7.3.4 Important properties

Evaluating guards Since ExecRun (Algorithm 10) performs a *symbolic* interpretation of the model, all the possible modes (valuations for the guards) must be hypothesized and explored at

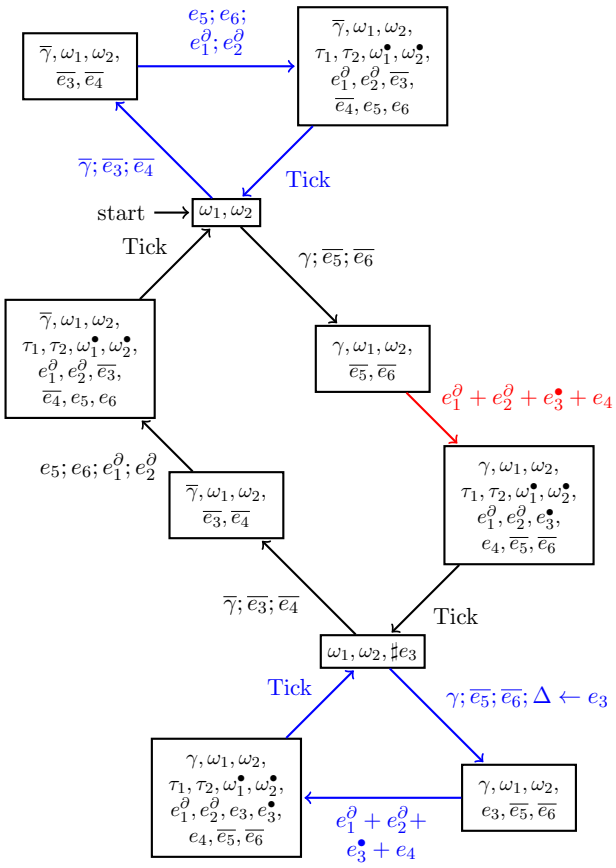


Figure 20: ExecRun as a labeled automaton for the Simple Clutch. Notations: For all statuses (shown in boxes), v (respectively $\overline{v}$) means $v = \text{T}$ (respectively $v = \text{F}$), and not mentioning v means $v = \text{U}$. $\sharp e$ means that e_f belongs to context Δ. The assignment $\Delta \leftarrow e_3$ refers to Line 10 of Algorithm 10. Blue (respectively black) transitions belong to a continuous-time (respectively discrete-time) dynamics. The red transition is impulsive. A semicolon is the sequential composition of computations, and the + symbol defines blocks of equations.

Line 10.[20] The program is accepted if and only if, for all possible modes, no *Fail* is returned by ExecRun. In case of success, all reachable pairs (σ, Δ) of statuses and contexts have been explored.

ExecRun as a labeled automaton We can see ExecRun as a labeled finite state automaton having nodes labeled with the different encountered status-context pairs, and transitions labeled with the micro-steps. In Figure 20, we depict this graph for the clutch example. The Tick transitions indicate a move to the next nonstandard instant.

Rejecting models Firstly, a system exhibiting logico-numerical fixpoint equations is rejected via synctactic checks, since it does not agree with Definition 33. Then, a failure is found in the following cases:

- SigmaMethod (Algorithm 6) itself fails, indicating that no matching is found for the bipartite graph $\mathcal{G}_G$ of the running dAE system G=0, a failure criterion for the Σ-method. Failure message indicates either over- or under-determination.
- The conflict between the predictions from the past and the consistency conditions from the current mode cannot be solved, i.e., $\beta_{\mathfrak{U}} \neq \emptyset$ at Line 5 of SolveConflict, revealing some missing

[20] Assertions may be used to restrict the set of possible modes, by stating some physical knowledge the programmer may have, e.g., 'this sequence of modes is not possible'. This is a minor change to our approach, not developed here.

restart equation(s) for the new mode.

Detecting continuous modes Elementary cycles of ExecRun capture runs with stationary valuations of the guards and define the continuous dynamics in each mode. Other runs capture mode changes and their reset actions. During runs with constant valuations of the guards, we have $(G_\Sigma \cup \overline{G}_\Sigma) \setminus \Phi = G_\Sigma$. Hence, Line 1 returns $\beta_{\mathfrak{U}} = \beta_{\mathfrak{D}} = \emptyset$ and, thus, $\beta_{\mathfrak{E}} = G_\Sigma$: we recover the usual structural analysis for DAE systems.

Solving conflicts at mode changes takes only finitely many nonstandard instants This follows from the fact that, at a mode change, the maximal shifting degree of context Δ is finite and the maximal shifting degree of the conflicting subsystem decreases by one at each subsequent time step. See Section 3.3 on the Cup-and-Ball example as an illustration of this.

Determinism of the algorithm All the algorithms called throughout the execution of Algorithm 10 possess a unique return (at Line 3 and Line 6). Since Algorithm 10 involves no choice by itself, it is deterministic. Comment 18 following Lemma 17 finds its justification here: our design choice in Lemma 17 was essential in guaranteeing the determinism of our compilation method.

7.3.5 Generic Blocks within a long mode

The results of this section will be used in Section 10, and more specifically in Section 10.4. The successive application of Line 3 and Line 6 of Algorithm 10 yields blocks of a specific form, that is investigated next.

Notations 41 For $f : \mathbb{R}^K \to \mathbb{R}$ a smooth numerical function and $n \geq 0$ an integer, set

$$\begin{aligned} B_\bullet(f,n) &=_{\text{def}} \begin{bmatrix} B^{\mathsf{h}}_\bullet(f,n) \\ B^{\mathsf{t}}_\bullet(f,n) \end{bmatrix} \text{ where} \\ B^{\mathsf{h}}_\bullet(f,n) &=_{\text{def}} \begin{bmatrix} f \\ f^\bullet \\ \vdots \\ f^{\bullet(n-1)} \end{bmatrix} \text{ and} \\ B^{\mathsf{t}}_\bullet(f,n) &=_{\text{def}} f^{\bullet n} \ . \end{aligned} \tag{83}$$

$B_\bullet(f,n)$ is called an (f,n)-*block*, $B^{\mathsf{h}}_\bullet(f,n)$ is its *head* and $B^{\mathsf{t}}_\bullet(f,n)$ is its *tail*. By convention, $B_\bullet(f,n)$ is empty for $n < 0$ and $B^{\mathsf{h}}_\bullet(f,n)$ is empty for $n = 0$. For $\{f_1, \ldots, f_k, \ldots, f_K\}$ a tuple of K smooth functions in K dependent variables, define

$$F = \begin{bmatrix} f_1 \\ \vdots \\ f_K \end{bmatrix}, \quad N = \begin{bmatrix} n_1 \\ \vdots \\ n_K \end{bmatrix}, \quad B_\bullet(F,N) = \begin{bmatrix} B_\bullet(f_1,n_1) \\ \vdots \\ B_\bullet(f_K,n_K) \end{bmatrix} \tag{84}$$

and the definitions of $B^{\mathsf{h}}_\bullet(F,N)$ and $B^{\mathsf{t}}_\bullet(F,N)$ follow accordingly. □

Lemma 42 *Assume that the system is within a long mode. Then, successively applying* Line 3 *and* Line 6 *of* Algorithm 10 *to the system* $G{=}0$ *returns, either 'Fail', or a structurally nonsingular system* $H{=}B^{\mathsf{t}}_\bullet(G,N)$ *of the form* (103).

Proof Consider Line 3 of Algorithm 10. Since the system is in a long mode, SigmaMethod returns, either *Fail*, or $(G_\Sigma, \bar{G}_\Sigma) = (B^{\mathsf{t}}_\bullet(G,N), B^{\mathsf{h}}_\bullet(G,N))$, where G is the subset of enabled equations at status σ.

Move to Line 6. Since the system is in a long mode, we have $\bar{G}_\Sigma{=}\Phi$, expressing that consistency conditions are guaranteed by previous nonstandard instants. Hence, the Dulmage-Mendelsohn decomposition performed at Line 6 simply returns $\beta_{\mathfrak{E}}{=}G_\Sigma{=}B^{\mathsf{t}}_\bullet(G,N)$. □

7.4 Structural Analysis: general case

So far, we restricted ourselves to systems having only long modes. In this section, we consider the general case, where transient modes can also occur, possibly in cascades. The Cup-and-Ball example of Section 3 can be kept in mind while reading this section.

Our main task is to revisit index reduction. So far, this was performed by invoking the Σ-method, which requires the new mode to last for long enough to allow for an unlimited number of time shifts to be applied. If the current mode is transient, this no longer holds and the continuation of the system dynamics is determined by its successive future modes—see the analysis of the Cup-and-Ball example in Section 3.5.

Whereas the Σ-method applies to time-invariant systems only, the method of *differentiation arrays* (see Section 3.1 of [21]) can be adapted to time-varying systems.

7.4.1 Index reduction with Difference Arrays

The index reduction algorithm proposed in this section applies to time-varying DAE systems. It is, however, less computationally efficient than the Σ-method (when the latter can be applied).

Difference arrays for time-varying systems

In the following, $\mathbb{T}$ denotes a discrete-time index set possessing a minimal element, denoted by 0.

Definition 43 *A* time-varying dAE system *over* $\mathbb{T}$ *is a pair* $(X, \mathcal{F})$*, where* X *is a tuple of numerical variables taking its values in an Euclidian space* D*, and* $\mathcal{F} : \mathbb{T} \to \mathbb{E}$ *maps every instant* $t \in \mathbb{T}$ *to a dAE system* $F_t = 0$ *whose set of dependent variables is contained in* X*. A* solution *of* $(X, \mathcal{F})$ *is a trajectory* $\mathbf{x} : \mathbb{T} \to D$ *such that, for every* $t \in \mathbb{T}$*,* $\mathbf{x}(t)$ *satisfies* $F_t = 0$*.*

Example 44 Consider $\mathbb{T} = \mathbb{N}$, $X = (u, v, w)$, and $\mathcal{F}$ is defined by

$$t{=}0 : \ 0 = v^\bullet - 3u \ \ ; \ \text{and} \ \forall t > 0 : \begin{cases} 0 = u^\bullet - v \\ 0 = v - uw \\ 0 = w^\bullet + u \end{cases} .$$

The solutions of this system are trajectories $t \mapsto (u(t), v(t), w(t))$ such that

$$0 = v(1) - 3u(0) \ \ ; \ \ \forall t > 0 : \begin{cases} 0 = u(t+1) - v(t) \\ 0 = v(t) - u(t)w(t) \\ 0 = w(t+1) + u(t) \end{cases} .$$

Note that solutions are only partially specified at $t = 0$ since w is not involved in the initial conditions. □

Given a time-varying dAE system $(X, \mathcal{F})$, we consider, for each $k \in \mathbb{T}$, the following *dAE array*:

$$\mathcal{A}_k \ =_{\text{def}} \ \begin{bmatrix} F_0 \\ F_1 \\ \vdots \\ F_k \end{bmatrix} . \tag{85}$$

This array allows us to perform the 'index reduction' of F_0, by adding 'latent equations' taken from the future systems F_1, F_2, etc., generally different from $F_0^\bullet, F_0^{\bullet 2}$, etc. To this end, the set of all variables of array $\mathcal{A}_k$ decomposes as $Y \cup X \cup W$, where:

$$\begin{array}{l} Y \text{ collects the variables in } F_0 \text{ that participate in the consistency;} \\ X \text{ collects the leading variables of } F_0; \\ W \text{ collects all the remaining variables of the array.} \end{array} \tag{86}$$

In its Section 3.1 on 'standard indices', Campbell and Gear's landmark paper [21] states that, for time-invariant systems ($F_t \equiv F_0$), the index is the smallest k such that $\exists W: \mathcal{A}_k(X, W, Y)=0$ uniquely defines X as a function of Y, provided that Y is consistent.

Therefore, we consider the following structural translation of the above problem (compare with Problem 23 in Section 6.2):

Problem 45 *Find the smallest integer k such that the system $\exists W : \mathcal{A}_k(X, W, Y)=0$ is structurally nonsingular in the sense of Definition 21.*

For time-invariant systems ($F_t \equiv F$ for all t), Problem 45 is monotonic in that, if this problem has a solution for k, then it also has a solution for $k' > k$. However, this no longer holds for time-varying systems, since it can be that system $F_{k+1} = 0$ is in conflict with $\mathcal{A}_k$.

Handling finite cascades of transient modes

Now, we specialize the structural analysis of difference arrays to the handling of finite cascades of transient modes. That is, we assume the following finite chain of modes:

$$\underbrace{\mu_0, \mu_1 \ldots, \mu_K}_{\text{finite cascade of transient modes}} \quad , \quad \underbrace{\mu_\infty}_{\text{final long mode}} \tag{87}$$

We adopt the following principles in developing our structural analysis:

- For the final long mode μ_∞, it is legitimate to postpone the satisfaction of some of the consistency equations. That is, we reuse the conflict solving approach of the algorithm SolveConflict (Algorithm 9).
- For the transient modes $\mu_0, \mu_1 \ldots, \mu_K$, however, this is not legitimate, so that we require the corresponding equations to be satisfied and refuse to erase any of them.

We associate to the chain (87) the array

$$\mathcal{A} =_{\text{def}} \begin{bmatrix} F_0 \\ F_1 \\ \vdots \\ F_K \\ F_{\infty,\Sigma} \cup \overline{F}_{\infty,\Sigma} \end{bmatrix} \tag{88}$$

with a decomposition of its variables as $Y \cup X \cup W$ following (86).

The last row of this array originates from applying index reduction on the long mode μ_∞.

Based on the results of Section 6.1.4, in particular Lemma 22, the structural analysis for time-varying systems is described in Algorithm 11 (DiffArray). This algorithm requires as inputs an array $\mathcal{A}$ as in (88) and a status; it returns, either a *Fail* information ($b = \text{F}$), or a pair $(F_\Sigma, \overline{F}_\Sigma)$, where F_Σ=0 is a structurally nonsingular system extracted from $\mathcal{A}$ and determining the leading variables of F_0, and $\overline{F}_\Sigma$=0 is a solvable system of consistency equations. The algorithm proceeds according to two phases.

In a first phase (Line 1–Line 10), the subchain of transient modes is explored, with increasing array size $k = 0, \ldots, K$. Using status σ, the variables of $\mathcal{A}_k$ are sorted at Line 5 following (86). At Line 6, array $\mathcal{A}_k$ is submitted to ExistQuantifEqn (Algorithm 5), which returns the two Boolean flags $b_\mathfrak{O}$, indicating whether the overdetermined block is empty, and $b_\mathfrak{U}$, indicating whether the underdetermined block is empty. If $b_\mathfrak{O} = \text{F}$, which indicates an overdetermined system, a failure indication is returned at Line 7. If $b_\mathfrak{O} = \text{T}$ and $b_\mathfrak{U} = \text{F}$ is returned, the array is increased (Line 3) and reprocessed the same way, until $k = K$ is reached. Otherwise, the first phase returns the pair $(F_\Sigma, \overline{F}_\Sigma)$. The status is updated as $\text{I} \nearrow \text{U}$ for each latent equation added, and the algorithm successfully terminates.

Algorithm 11 DiffArray

Require: $(\mathcal{A}, \sigma)$ as in (88); **return** $(\sigma, b, F_\Sigma, \overline{F}_\Sigma)$

1: $k = -1$;
2: **if** $k < K$ **then**
3: $k \leftarrow k{+}1$
4: $\mathcal{A}_k \leftarrow (F_0, \ldots, F_k)$
5: arrange the variables of $\mathcal{A}_k$ as $(Y_k \cup X_k \cup W_k)$
6: $(b_\mathfrak{D}, b_\mathfrak{U}, F_\Sigma, \overline{F}_\Sigma) \leftarrow \mathsf{ExistQuantifEqn}(\mathcal{A}_k)$
7: **if** $\neg b_\mathfrak{D}$ **then** return $b \leftarrow$ F
8: **else**
9: **if** $\neg b_\mathfrak{U}$ **then go to** Line 2
10: **else** return $(b \leftarrow$ T$, F_\Sigma, \overline{F}_\Sigma)$; increase σ
11: **else** consider $\mathcal{A}$ defined in (88)
12: $\mathcal{A} \leftarrow \mathcal{A}$ where $\overline{F}_{\infty,\Sigma} \leftarrow (\overline{F}_{\infty,\Sigma} \setminus \mathcal{A}_K)$
13: $(\beta_\mathfrak{U}, \beta_\mathfrak{E}, \beta_\mathfrak{D}) = \mathrm{DM}(\mathcal{A})$
14: **if** $\beta_\mathfrak{D} \neq \emptyset$ **then**
15: $\mathcal{A} \leftarrow \mathcal{A}$ where $\overline{F}_{\infty,\Sigma} \leftarrow \overline{F}_{\infty,\Sigma} \setminus \beta_\mathfrak{D}$;
16: **go to** Line 13
17: **else**
18: **if** cond. 2 and 3 of Lemma 22 hold **then**
19: partition $\beta_\mathfrak{E} = F_\Sigma \cup \overline{F}_\Sigma$
20: return $(b \leftarrow$ T$, F_\Sigma, \overline{F}_\Sigma)$; increase σ
21: **else** return $b \leftarrow$ F

If $k{=}K$ is reached with no success, we move to the second phase, starting with the removal of duplicates between $\mathcal{A}_K$ and $\overline{F}_{\infty,\Sigma}$ (Line 12). Then, we apply the DM decomposition at Line 13 to identify possible conflicts and underdetermined parts. We handle conflicts as in the SolveConflict algorithm (Algorithm 9), by deleting conflicting equations from $\overline{F}_{\infty,\Sigma}$ (Line 15) and returning to Line 13; this loop is visited only once thanks to Lemma 17. When reaching Line 17, we know that condition 1 of Lemma 22 is satisfied, so that we can conclude by using the successful case of Lemma 22 at Lines 18 and subsequent. The status is updated as I $\nearrow$ U for each latent equation added (Line 20).

7.4.2 Executing a nonstandard instant

We first revisit the algorithm IndexReduc. If the current mode is long, then IndexReduc is implemented as Algorithm 8. Otherwise, we will implement it by hypothesizing a future mode trajectory and calling the algorithm DiffArray (Algorithm 11). IndexReduc, revisited, is specified as Algorithm 12.

Algorithm 12 IndexReduc, revisited

Require: $(G, \sigma, \mathit{type})$; **return** $(\sigma, b, G_\Sigma, \overline{G}_\Sigma)$

1: **if** *type*=long **then**
2: $(b, G_\Sigma, \overline{G}_\Sigma) \leftarrow \mathsf{SigmaMethod}(G)$ with dependent variables set by σ; increase σ
3: **else**
4: hypothesize $\mathcal{F} = (G_0, G_1, \ldots)$ with $G_0{=}G$; let $\mathcal{A}$ be the corresponding array
5: $(b, G_\Sigma, \overline{G}_\Sigma) \leftarrow \mathsf{DiffArray}(\mathcal{A}, \sigma)$; increase σ
6: **if** b **then return** $(G_\Sigma, \overline{G}_\Sigma)$

In this algorithm, Line 4 requires hypothesizing a continuation for the current mode. This, of course, is problematic if all modes can be visited in this continuation, particularly transient modes. It is therefore essential to be able, at compile time, to restrict the number of possible continuations

for the current mode. This question is investigated in Section 7.4.3.

Algorithm 13 ExecRun , revisited

Require: (σ, Δ); **return** $(\mathit{Fail}, \sigma, (\sigma^\circ, \Delta^\circ))$

1: $\mathit{type} \leftarrow \mathsf{EvalModeType}(\sigma)$
2: $\Phi \leftarrow \{e \in \Delta \mid \sigma(\gamma(e)) = \mathrm{T}\}$
3: $G \leftarrow [\mathit{Enab}(\sigma) \cap \mathit{Undef}(\sigma)]$
4: $(\sigma, b, G_\Sigma, \overline{G}_\Sigma) \leftarrow \mathsf{IndexReduc}(G, \sigma, \mathit{type})$; increase σ
5: **if** $\neg b$ **then return** $\mathit{Fail}(\sigma)$
6: **else**
7: $\quad(\sigma, b, H) \leftarrow \mathsf{SolveConflict}((G_\Sigma \cup \overline{G}_\Sigma) \setminus \Phi, \sigma)$; increase σ
8: $\quad$**if** $\neg b$ **then return** $\mathit{Fail}(\sigma)$
9: $\quad$**else**
10: $\quad\quad\sigma \leftarrow \mathsf{Eval}(\sigma, H)$; increase σ
11: $\quad\quad(\sigma^\circ, \Delta^\circ) \leftarrow \mathsf{Tick}(\sigma)$

Having revisited index reduction, we can now revisit ExecRun, see Algorithm 13. Only two lines are modified with reference to the original algorithm ExecRun:

Line 1 We add the evaluation of the *type* (long or transient) of the current mode.

Line 4 We call instead the IndexReduc corresponding to Algorithm 12, which selects the appropriate method for index reduction, depending on the *type* of the current mode.

7.4.3 Continuations of transient modes

We need to investigate this issue for transient modes only. Therefore, we begin with assumptions regarding transient modes. Recall that we follow Notations 26, hence, by *numerical variable* of an mDAE system S, we mean a variable of the form x'^m (m-th derivative of x) for some algebraic or state variable x of the considered system. The *degree* of x'^m is m. The same holds regarding mdAE system S, with $x^{\bullet m}$ (m-shifted version of x) replacing x'^m. We repeatedly use these notations in the sequel.

Assumptions regarding transient modes According to Definition 31, a mode μ is transient if at least one of its T-valued guards is transient. Guards are Boolean expressions of predicates over numerical variables. We will restrict ourselves to the class of nonstandard mdAE models S with set X of numerical variables, satisfying the following assumption regarding transient predicates over numerical variables:

Assumption 3 *Let $\gamma_1, \gamma_2 \in \Gamma(S)$ be any two distinct transient predicates over numerical variables of mdAE model S. Then, for ν_X^1 and ν_X^2 any two valuations of the tuple X such that $\nu_X^1 - \nu_X^2$ is infinitesimal, $\gamma_1(\nu_X^1) = \gamma_2(\nu_X^2) = \mathrm{T}$ cannot occur.*

Discussion. Assumption 3 formalizes that two different transient predicates never take the value T simultaneously, "by chance". For example, for $x, y \in X$, the two predicates $\gamma_1 =$ `when` $x \geq 0$ and $\gamma_2 =$ `when` $y \leq 5$ specify the events when x becomes ≥ 0 and when y becomes ≤ 5. The assumption says that these two events will never occur simultaneously. For this case, Assumption 3 is very reasonable. Of course, if $\gamma_1 =$ `when` $x^3 \geq 8$ and $\gamma_2 =$ `when` $x \geq 2$, the two guards are considered different (their syntax differ) but generate simultaneous events. It is not the duty of the compiler to discover if two guards happen to be mathematically identical whereas they differ in their syntax. It is rather the responsibility of the programmer to use identical syntax for different occurrences of a same predicate. □

Using Assumption 3 we can in particular bound the number of transient modes that can occur at a given instant $t \in \mathbb{T}$. For $\gamma_o \in \Gamma(S)$ a transient predicate, let

$$\Gamma(S, \gamma_o) \quad \subseteq \quad \Gamma(S) \tag{89}$$

be the subset of all guards γ of S that are Boolean expressions of predicates such that $\gamma = \mathrm{T}$ requires $\gamma_o = \mathrm{T}$. An example would be $\gamma = \gamma_o \wedge \gamma'$. The following obvious lemma holds:

Lemma 46 *Under Assumption 3, the number of transient modes that can possibly occur at a given instant $t \in \mathbb{T}$ is bounded by*

$$M(S) \quad = \quad \max_{\gamma_o} \left| 2^{\Gamma(S,\gamma_o)} \right|$$

where γ_o ranges over the set of all transient predicate guards of S, and $|A|$ denotes the cardinal of set A.

In practice, this bound is generally much smaller than the number of possible modes for S, which is bounded by $|2^{\Gamma(S)}|$.

Assumptions regarding continuations of transient modes Let $\mu_0 = \mu(\mathfrak{t})$ be the current mode at time $\mathfrak{t} \in \mathbb{T}$, and let $\mu_0, \mu_1, \mu_2, \ldots$ be any continuation for μ_0 at instants $\mathfrak{t}, \mathfrak{t}^\bullet, \mathfrak{t}^{\bullet 2}, \ldots$. Such continuation can have one of the following two forms:

$$\text{finite cascade} \quad : \quad \mu_0, \mu_1, \ldots, \mu_{m-1}, \mu_\infty \tag{90}$$

$$\text{infinite cascade} \quad : \quad \mu_0, \mu_1, \ldots, \mu_{m-1}, \ldots \tag{91}$$

where the final mode μ_∞ in (90) is long (and thus repeated), and all other listed modes, including the "...", are transient. We will state assumptions that statically rule out (91) and limit the number of possible continuations in (90). For μ a transient mode, let

$$X(\mu) \subseteq X$$

be the subset of numerical variables x such that $x^\bullet - x$ is possibly non-infinitesimal at this mode. In other words, $X(\mu)$ is the set of variables that are reset by this transient mode. We then define, for M a set of transient modes:

$$X(M) = \bigcup_{\mu \in M} X(\mu) \ .$$

This set can be identified using structural algorithms.

Example 47 The statement

$$x' = f(x, u) \ \texttt{reset} \ 0 \ \texttt{when} \ x \geq 1$$

expands, according to Definition 27, as

$$\left\{ \begin{array}{lll} \texttt{when} \ x \geq 1 & \texttt{then} & x^+ = 0 \\ & \texttt{else} & x' = f(x, u) \end{array} \right.$$

whose nonstandard semantics is

$$\left\{ \begin{array}{lll} \texttt{when} \ x \geq 1 & \texttt{then} & x^\bullet = 0 \qquad (e_1) \\ & \texttt{else} & \frac{x^\bullet - x}{\partial} = f(x, u) \quad (e_2) \end{array} \right. \tag{92}$$

Since $x^\bullet - x$ is not guaranteed infinitesimal by (e_1), $x^\bullet - x$ is non-infinitesimal at the instant of zero-crossing (i.e., the first time when $x \geq 1$ occurs). Thus, $X(\mu_{\mathrm{zc}}) = \{x\}$, where μ_{zc} is the mode when the zero-crossing occurs. This can be found at compile time, based on the syntax of System (92).□

Example 48 Consider the Cup-and-Ball example with elastic impact, see Section 3.5, in its original specification prior to adding the restart condition (46). Let μ_{imp} be the mode when the elastic impact occurs. Then, $X(\mu_{\text{imp}}) = \emptyset$. Thus, we know that the only continuation is: $\mu_{\text{imp}}, \mu_{\text{free}}$, in which μ_{free} is the mode in which the rope is not straight and, thus, free motion occurs. The same holds after adding restart condition (46). □

In the sequel, we denote by M the set of all possible modes of system S. For $Y \subset X$ a subset of variables, let

$$M(Y) \subseteq M$$

be the set of all transient modes that can possibly be visited as a consequence of making noninfinitesimal changes to the variables of Y—this set can be identified at compile time using structural algorithms.

In bounding the possible continuations of a transient mode μ_0, we consider the following chain of sets of possibly reachable transient modes:

$$\begin{array}{rl} \mathsf{Continuations}\,(\mu_0) = & M_0, M_1, M_2, \ldots \\ \text{where}: & M_0 = \{\mu_0\} \\ \text{and } \forall k > 0: & M_k = M(X(M_{k-1})) \end{array}$$

Once again thanks to structural algorithms, the following assumption can be checked at compile time on the considered system S:

Assumption 4 *For every transient mode μ, the chain $X_k, k \geq 0$ in* $\mathsf{Continuations}\,(\mu)$ *converges to $\emptyset$ in finitely many steps.*

Lemma 49 *Assumption 5 implies Assumption 4,*

where Assumption 5 states:

Assumption 5 *For every subset Y of reset variables,*

$$d^o(X(M(Y))) \quad > \quad d^o(Y)$$

where $d^o(Z)$ is the smallest (shifting or differentiation) degree among all variables of Z.

Assumption 5 holds if, when a transient mode is reached due to a predicate on positions $x, y, \ldots$, then only velocities $x', y', \ldots$ or accelerations $x'', y'', \ldots$ can be reset. This kind of property is often encountered in classes of physical systems, e.g., contact mechanics.

8 Systems with shifts and differentiations

So far, our formalism of multi-mode DAE systems handles only guarded DAEs. For more elaborate examples, we need to support also left- and right-limits, to be able to explicitly define restart or reset values for the dynamics of a given mode. With reference to model class (1), our extension takes the form

$$\begin{array}{rl} \text{if} & \gamma_j(\text{the } x_i \text{ their derivatives and left-limits}) \\ \text{do} & f_j(\text{the } x_i \text{ their derivatives and right-limits}) = 0 \end{array} \tag{93}$$

Right-limits are typically used in defining restart values at mode changes.

Since we will map the dynamics in the nonstandard domain, we will represent the left- and right-limits x^- and x^+ through the backward and forward shifts ${}^\bullet x$ and $x^\bullet$, see Definition 2. We thus consider a syntax offering derivative $'$ and forward shift $\bullet$, both related by the following identities in the nonstandard semantics:

$$\begin{array}{rcl} x' & = & \frac{1}{\partial}(x^\bullet - x) \\ x^\bullet & = & x + \partial x' \end{array} \tag{94}$$

The resulting formalism will be called mdDAE, or ∂AE, for short. We formalize this next.

8.1 Preliminaries

In the nonstandard semantics, shift and derivative are related via identities (94). In these preliminaries we investigate additional relations and properties that follow from identities (94).

Lemma 50

1. *The two operators* $\bullet$ *and* $\prime$ *commute: for every variable* x*, we have* $x^{\bullet\prime} = x'^{\bullet}$.
2. *For* $f : \mathbb{R}^k \mapsto \mathbb{R}$ *a* $\mathcal{C}^1$*-function and* X *a vector variable of dimension* k*, define*

$$\begin{array}{rcl} f^{\bullet}(X) =_{\text{def}} [f(X)]^{\bullet} & =_{\text{def}} & f(X^{\bullet}) \\ (f(X))' & =_{\text{def}} & \nabla f(X).X' \end{array} \tag{95}$$

where ∇f *denotes the Jacobian of* f*. Then, differentiation and shift commute, meaning that:* $[f(X)]^{\bullet\prime} = [f(X)]'^{\bullet}$. □

Proof Statement 1 follows immediately from identities (94). Then, the equalities

$$\begin{array}{rcl} [f(X)]^{\bullet\prime} = [f^{\bullet}(X)]' & = & \nabla f(X^{\bullet}).X^{\bullet\prime} \\ \text{(by Statement 1)} & = & [\nabla f(X).X']^{\bullet} \\ & = & [f(X)]'^{\bullet} \end{array}$$

prove Statement 2. □

We consider the alphabet $\{\bullet, \prime\}$ related through identities (94), representing the shift and the derivative operators, respectively. Successive differentiations or shifts of a variable x are denoted by x^w for some word $w \in \{\bullet, \prime\}^*$ where * denotes the usual Kleene closure. We denote by $\bullet n$ and $\prime n$ the concatenation of n successive $\bullet$ and $\prime$, respectively. In x^w, word w is read from left to right, e.g., in $x'^{\bullet}$, x is first differentiated, and then x' is shifted. We denote by X^w the set of all x^w for x ranging over the set X of variables. For $\Sigma \subseteq \{\bullet, \prime\}^*$, we set

$$X^{(\Sigma)} =_{\text{def}} \bigcup_{w \in \Sigma} x^w \quad \text{and} \quad \{X\} =_{\text{def}} X^{(\{\bullet,\prime\}^*)} \tag{96}$$

Definition 51 *A* ∂AE system *is a finite set of* guarded equations *of the form:*

$$e \ ::= \ \texttt{if}\ \gamma\ \texttt{do}\ f{=}0 \tag{97}$$

$$\gamma^{\bullet} \ ::= \ \mathit{bexp} \tag{98}$$

where f *is a smooth scalar function over* $\{X\}$*, the shifted guard* $\gamma^{\bullet}$ *is defined by* $bexp$*, which is a* typed *boolean expression of predicates over* $\{X\}$*, and the guard* γ *is typed* `long/transient`.

The reader is invited to compare this definition with Definition 33. The notations of Definition 33 and the explanations that follow it extend to ∂AE. In a ∂AE model we may find $x, x', x^{\bullet}, x''^{\bullet}, x'^{\bullet\prime}$ and so on. Using (94) and Lemma 50 repeatedly, $x'^{\bullet\prime}$ can be expressed as a function of $x, x^{\bullet}, x^{\bullet 2}$, $x^{\bullet 3}$ or as a function of x, x', x'', x'^3. Consequently, for $w \in \{\prime, \bullet\}^*$ a word over the alphabet $\{\prime, \bullet\}$, it is natural to define the *degree* of x^w with respect to x as being $|w|$, the length of word w. For X, $\Sigma \subseteq \{\bullet, \prime\}^*$, and $X^{(\Sigma)}$ as in (96), let E be a system of algebraic equations over the set $X^{(\Sigma)}$.

$$\text{We denote by } {}^{(\bullet)}E \text{ the system obtained by expressing any monomial } x^w \text{ as a function of } x, x^{\bullet}, x^{\bullet 2}, x^{\bullet 3}, \ldots, \text{ by repeatedly using the first identity of (94) and Lemma 50.} \tag{99}$$

System ${}^{(\bullet)}E$ involves only shifts and no differentiations. Vice-versa, recalling (95):

$$\text{For } E \text{ of the shifted form } E = F^{\bullet K}, \text{ let } {}^{(\prime)}E \text{ be obtained by repeatedly applying, to } E, \text{ until all shifts get eliminated from it, the map } G(X^{\bullet}) \mapsto G(X) + \nabla G(X).X', \text{ where } \nabla G \text{ denotes the Jacobian of } G. \tag{100}$$

Note the difference in defining ${}^{(\bullet)}E$ and ${}^{(\prime)}E$.

8.2 Shifting versus differentiating

The whole Sections 7 and 7.3, where the structural analysis of mdAE systems was developed, extends verbatim to ∂AE systems, up to the following changes:

- Replace everywhere the superscript $\bullet m$, for $m\in\mathbb{N}$, by a word $w\in\{\prime,\bullet\}^*$ of length $|w|=m$; define, for $e^w :: \texttt{if}\ \gamma\ \texttt{do}\ f^w=0$, the operators

$$\begin{aligned}\mathsf{ForwardShift}: e^{w\bullet} &=_{\text{def}} \texttt{if}\ \gamma\ \texttt{do}\ f^{w\bullet}=0\\ \mathsf{Differentiate}: e^{w\prime} &=_{\text{def}} \texttt{if}\ \gamma\ \texttt{do}\ f^{w\prime}=0\end{aligned} \tag{101}$$

- When invoking the Σ-method (via IndexReduc) in Algorithm 10 (respectively Algorithm 13), we can either use differentiation or forward shift at Line 3 (respectively Line 4). The question is: does this choice influence the outcome of the Dulmage-Mendelsohn decomposition at Line 1 of SolveConflict (Algorithm 9)? Other steps of Algorithm 10 or Algorithm 13 are for sure not affected.

The alternative "shift or differentiate" raises the following question: *what is the effect, on the structural analysis, of chosing a particular alternative each time we reach Line 3 of Algorithm 10 or Line 4 of Algorithm 13*? We address this issue next.

8.3 Invariance results

We use the notion of weighted bipartite graph associated to a ∂AE system $F=0$. We refer the reader to Section 6.2 for this notion associated with DAE systems, and we extend it to ∂AE systems by deducing weights from the length $|w|$ of words over the alphabet $\{\bullet,\prime\}^*$. Thanks to the two-way correspondence (94), the following holds:

Lemma 52 *For $F=0$ any system of algebraic equations over $X^{(\Sigma)}$: F, ${}^{(\bullet)}F$, and ${}^{(\prime)}F$ (when the latter can be considered) possess identical weighted bipartite graphs. As a consequence, applying* Algorithm 6 *(the Σ-method) to F, ${}^{(\bullet)}F$, or ${}^{(\prime)}F$ yields the same result in terms of offsets and outcome of the DM decomposition.*

Recall the following definition: An *execution tree* for Algorithm 10 is a labeled tree T representing all its possible runs. Nodes of T are labeled by one of the 10 action labels of Algorithm 10 and each maximal path of T describes a possible run of Algorithm 10 through the sequence of action labels sitting at the nodes of this path. In this section, we investigate the effect of

selecting one of the two alternatives in (101), each time index reduction is performed. (102)

Performing (102) results in corresponding variations of the context Δ at the next instant. We thus consider the set $\mathcal{T}$ of all executions trees obtained by performing (102) in Algorithm 10. From Lemma 52 we immediately deduce:

Lemma 53 *$\mathcal{T}$ is a singleton consisting of an execution tree that we denote by T_S. In addition, the contribution, to the bipartite graph, by the context Δ, is independent from these choices. The same holds for* Algorithm 13.

Lemma 53 means that Algorithms 10 and 13 will visit the same sequence of action labels, irrespectively of all the choices made (either shifting or differentiating). The systems of equations produced by Line 3 of Algorithm 10 and Line 4 of Algorithm 13 may differ, but possess identical weighted bipartite graphs. From now on we focus on Algorithm 13, of which Algorithm 10 is a specialization.

8.4 Executing a nonstandard instant, revisited

In Algorithm 13, forward shifting is used when performing index reduction. In Section 8, however, we have seen that, in Algorithm 13 forward shifting and differentiation can be substituted for one another. This leads us to consider the following strategy, whose motivation is to make standardization easier (see the standardisation of the Cup-and-Ball example in Section 3.4):

Strategy 54 *In* Algorithm 13*, we priorize shifts or differentiations depending on the following cases:*

1. *The considered nonstandard instant belongs to a cascade of events: we priorize shifts;*
2. *The considered nonstandard instant is in a continuous mode: we priorize differentiation;*
3. *We are entering a continuous mode: we priorize differentiation unless it brings leading derivatives.*

The justification of point 2 is clear: we should reproduce the usual index reduction. In a cascade of mode changes occurring at successive nonstandard instants, we are in a discrete time dynamics and thus point 1 follows. Finally, point 3 is justified by the fact that, if x'^n is the leading derivative for x in the new mode, we need to know restart values for the lower derivatives $x, x', \ldots, x'^{(n-1)}$, given by the values for the variables $x^\bullet, x'^\bullet, \ldots, x'^{(n-1)\bullet}$.

8.5 Generic Blocks

Despite all the different choices (either $'$ or $\bullet$) give rise to the same execution tree T_S by Lemma 53, they may result in different standardizations. We thus want to refine our understanding of the effect of choices (102) when applying Algorithm 13. What we are really interested in is what happens at Line 10 where enabled blocks are solved: it may matter whether shift or differentiation was used at each particular traverse of Line 4 of Algorithm 13. To this end we investigate the effect, on a given $e \in \mathbb{E}^S$, of the choices (102) performed in Algorithm 13, where $\mathbb{E}^S$ is the set of all the guarded equations of a given ∂AE system S. The successive application of Line 4 and Line 7 of Algorithm 13 yields blocks of a specific form that we investigate next.

Notations: Let $w \in \{', \bullet\}^*$ be a word of length n, let $w[0], \ldots, w[n-1]$ be the successive prefixes of w, starting from the empty prefix. For $f : \mathbb{R}^K \mapsto \mathbb{R}$ a smooth numerical function and w as above, set

$$B(f,w) \quad =_{\text{def}} \quad \begin{bmatrix} B^{\mathsf{h}}(f,w) \\ B^{\mathsf{t}}(f,w) \end{bmatrix} \quad \text{where}$$

$$B^{\mathsf{h}}(f,w) \quad =_{\text{def}} \quad \begin{bmatrix} f^{(w[0])} \\ f^{(w[1])} \\ \vdots \\ f^{(w[n-1])} \end{bmatrix} \quad \text{and}$$

$$B^{\mathsf{t}}(f,w) \quad =_{\text{def}} \quad f^w$$

$B(f,w)$ is called an (f,w)-*block*, $B^{\mathsf{h}}(f,w)$ is its *head* and $B^{\mathsf{t}}(f,w)$ is its *tail*. For $\{f_1, \ldots, f_k, \ldots, f_K\}$ a tuple of K smooth functions in K dependent variables, $w_1, \ldots, w_k, \ldots, w_K \in \{', \bullet\}^*$ a set of K words of respective lengths $n_1, \ldots, n_k, \ldots, n_K$, define

$$F = \begin{bmatrix} f_1 \\ \vdots \\ f_K \end{bmatrix}, W = \begin{bmatrix} w_1 \\ \vdots \\ w_K \end{bmatrix}, B(F,W) = \begin{bmatrix} B(f_1,w_1) \\ \vdots \\ B(f_K,w_K) \end{bmatrix} \tag{103}$$

and the definitions of $B^{\mathsf{h}}(F,W)$ and $B^{\mathsf{t}}(F,W)$ follow accordingly. In addition, we will write

$$B^{\mathsf{h}}_{\bullet}(f,n) =_{\text{def}} B^{\mathsf{h}}(f,\bullet n) \quad \text{and} \quad B^{\mathsf{h}}_{\prime}(f,n) =_{\text{def}} B^{\mathsf{h}}(f,\prime n)$$

□

Exchanging $\prime \leftrightarrow \bullet$: In this paragraph we detail such exchanges for the above identified blocks. Let $f : \mathbb{R}^m \mapsto \mathbb{R}$ be a function and let $f(X)=0$ be the associated equation with m dependent variables collected in the vector X. Consider $w\in\{\prime,\bullet\}^*$, and let

$$f^w \quad \mapsto \quad \left(f^{(\prime n)}, f^{(\bullet n)}\right), \text{ where } n = |w|, \tag{104}$$

be the map defined by the following algorithm:

1. Initialize $h_0 = g_0 := f^w$ and $v_0 := w$;
2. For $j = 1,\ldots,n$:
   (a) Decompose $v_{j-1} := v_j.\nu$ where $\nu \in \{\prime,\bullet\}$;
   (b) **case $\nu = \prime$:** define $h_j := h_{j-1}$ and

$$g_j \quad := \quad g_{j-1}\left[x' \leftarrow \tfrac{x^{\bullet}-x}{\partial}\right], \tag{105}$$

   where x' ranges over the set of derivatives involved in g_{j-1};
   case $\nu = \bullet$: h_{j-1} has the form $\ell^{\bullet}(Y)$ for some function h and Y its set of dependent variables; define $g_j := g_{j-1}$ and

$$h_j(Y) \quad := \quad \nabla\ell(Y).Y', \tag{106}$$

   where $\nabla\ell$ is the Jacobian of ℓ.
3. Return $\left(f^{(\prime n)}, f^{(\bullet n)}\right) = (h_n, g_n)$.

Lemma 55 *Applying map (104) to all the entries of $B^{\mathsf{h}}(f,w), B^{\mathsf{h}}(f,w)$, and $B^{\mathsf{t}}(f,w)$, returns*

$$\begin{array}{l} B^{\mathsf{h}}_{\prime}(f,n) \quad \textit{and} \quad B^{\mathsf{t}}_{\prime}(f,n), \\ B^{\mathsf{h}}_{\bullet}(f,n) \quad \textit{and} \quad B^{\mathsf{t}}_{\bullet}(f,n), \end{array} \tag{107}$$

respectively, where $n = |w|$. Doing the same for $B(F,W)$ defined in (103) returns

$$B_{\prime}(F,N) \quad \textit{and} \quad B_{\bullet}(F,N) \tag{108}$$

where N is the vector of integers collecting the lengths of the entries of W.

Comment 56 By regarding all x', x'', etc. as dummy variables,[21] every structurally nonsingular block $B(F,W) = 0$ defines a dAE system over nonstandard time basis $\mathbb{T}$. This also holds for $B_{\prime}(F,N) = 0$ and $B_{\bullet}(F,N) = 0$. On the other hand, by not involving shifts, $B_{\prime}(F,N) = 0$ defines a (continuous time, standard) DAE system. This is not true for $B(F,W) = 0$ in general. □

This finishes our study of structural analysis for mDAE or mdAE systems. Referring to Figure 18 in Section 5, we are ready to move to the two downgoing arrows, consisting in the standardization of the nonstandard model resulting from the structural analysis, together with the impulse analysis. The rest of the paper is devoted to this.

[21] This means that we regard x', x'', etc. as different variable names and forget about the fact that they are related via differentiation. The same term holds whith shifting.

9 Background on nonstandard analysis

The development of the clutch example in Section 2 consisted of three main steps: (1) the mapping of the original clutch model to its nonstandard semantics, (2) the symbolic execution of this nonstandard semantics (Section 2.4), and (3) the back-standardization of this symbolic execution (Section 2.5). All of this resulted in the actual code for the simulation results reported in Figure 9. Formally justifying back-standardization requires a non-trivial use of nonstandard analysis, way beyond the quick and superficial introduction we gave in Section 2.3. In this and the next sections, we provide the mathematical background supporting back-standardization procedures in full generality.

Nonstandard analysis was proposed by Abraham Robinson in the 1960s to allow for the explicit manipulation of "infinitesimals" in analysis [49, 23]. Robinson's approach was axiomatic, by adding three new axioms to the basic Zermelo-Fraenkel (ZFC) framework. An alternative presentation was later proposed by Lindstrøm [35]. Its interest is that it does not require any fancy axiomatic material but only makes use of Zorn's lemma, equivalent to the axiom of choice in the ZF set theory. The proposed construction bears some resemblance to the construction of $\mathbb{R}$ as the set of equivalence classes of Cauchy sequences in $\mathbb{Q}$ modulo the equivalence relation $\sim$ defined by $(u_n) \sim (v_n)$ iff $\lim_{n\to\infty}(u_n - v_n) = 0$.

The important point for us is that nonstandard analysis allows the use of the nonstandard discretization of continuous dynamics "as if" it was operational and with infinitesimal discretization error. Iwasaki et al. [33] first proposed using nonstandard analysis to discuss the nature of time in hybrid systems. Bliudze and Krob [14, 13] have also used nonstandard analysis as a mathematical support for defining a system theory for hybrid systems. In their study of mathematical foundations of hybrid systems modelers of the ODE class, Benveniste et al. [5] used extensively the nonstandard semantics of hybrid systems, with practical consequences for how to design a specification formalism for multimode ODE systems [15, 4].[22] See also [51, 39, 29] for similar studies using the axiomatic approach to nonstandard analysis. The following presentation is borrowed from [5].

9.1 Intuitive introduction

We begin with an intuitive introduction to the construction of the nonstandard reals. The goal is to augment $\mathbb{R} \cup \{\pm\infty\}$ by adding, to each x in the set, a set of elements that are "infinitesimally close" to it. We will call the resulting set ${}^\star\mathbb{R}$. Another requirement is that all operations and relations defined on $\mathbb{R}$ should extend to ${}^\star\mathbb{R}$.

A first idea is to represent such additional numbers as convergent real sequences. For example, elements infinitesimally close to the real number zero are the sequences $u_n = 1/n$, $v_n = 1/\sqrt{n}$ and $w_n = 1/n^2$. Observe that the above three sequences can be ordered: $v_n > u_n > w_n > 0$ where 0 denotes the constant zero sequence. Of course, infinitely large elements (close to $+\infty$) can also be considered, e.g., sequences $x_u = n$, $y_n = \sqrt{n}$, and $z_n = n^2$.

Unfortunately, this way of defining ${}^\star\mathbb{R}$ does not yield a total order since two sequences converging to zero cannot always be compared: if u_n and u'_n are two such sequences, the three sets $\{n \mid u_n > u'_n\}$, $\{n \mid u_n = u'_n\}$, and $\{n \mid u_n < u'_n\}$ may even all be infinite. The beautiful idea of Lindstrøm is to enforce that *exactly one of the above sets is important and the other two can be ignored.*

The key step in Lindstrøm's construction consists in fixing once and for all a finitely additive positive measure μ over the set $\mathbb{N}$ of integers with the following properties:[23]

$$\begin{aligned}&\mu : 2^{\mathbb{N}} \to \{0, 1\}\\ &\mu(X) = 0 \text{ whenever } X \text{ is finite}\\ &\mu(\mathbb{N}) = 1\end{aligned}$$

Once μ is fixed, one can compare any two sequences u_n and u'_n as follows. Exactly one of the three sets $\{n \mid u_n > u'_n\}$, $\{n \mid u_n = u'_n\}$, or $\{n \mid u_n < u'_n\}$ has μ-measure 1 whereas the other two must

[22] Corresponding to the Simulink class of modeling languages, with no DAE.

[23] The existence of such a measure is non-trivial and is explained later.

have μ-measure 0. Thus, we say that

$$\begin{array}{lll} u > u' & \text{if} & \mu(\{n \,|\, u_n > u'_n\}) = 1 \\ u = u' & \text{if} & \mu(\{n \,|\, u_n = u'_n\}) = 1 \\ u < u' & \text{if} & \mu(\{n \,|\, u_n < u'_n\}) = 1 \end{array}$$

respectively. Indeed, the same trick works for many other relations and operations on nonstandard real numbers, as we shall see. We now proceed with a more formal presentation.

9.2 Nonstandard domains

For I an arbitrary set, a *filter* $\mathcal{F}$ over I is a family of subsets of I such that: (1) the empty set does not belong to $\mathcal{F}$; (2) $P, Q \in \mathcal{F}$ implies $P \cap Q \in \mathcal{F}$; and (3) $P \in \mathcal{F}$ and $P \subset Q \subseteq I$ implies $Q \in \mathcal{F}$. Consequently, $\mathcal{F}$ cannot contain both a set P and its complement P^c. A filter that contains one of the two for any subset $P \subseteq I$ is called an *ultra-filter*. At this point we recall Zorn's lemma, known to be equivalent to the axiom of choice:

Lemma 57 (Zorn's lemma) *Any partially ordered set $(X, \leq)$ such that any chain in X possesses an upper bound has a maximal element.*

A filter $\mathcal{F}$ over I is an ultra-filter if and only if it is maximal with respect to set inclusion. By Zorn's lemma, any filter $\mathcal{F}$ over I can be extended to an ultra-filter over I. Now, if I is infinite, the family of sets $\mathcal{F} = \{P \subseteq I \mid P^c \text{ is finite}\}$ is a *free* filter, meaning it contains no finite set. It can thus be extended to a free ultra-filter over I. Hence:

Lemma 58 *Any infinite set has a free ultra-filter.*

Every free ultra-filter $\mathcal{F}$ over I uniquely defines, by setting $\mu(P) = 1$ if $P \in \mathcal{F}$ and 0 otherwise, a finitely additive measure[24] $\mu : 2^I \mapsto \{0, 1\}$, which satisfies

$$\mu(I) = 1 \text{ and, if } P \text{ is finite, then } \mu(P) = 0. \tag{109}$$

Now, fix an infinite set I and a finitely additive measure μ over I as above. Let $\mathbb{X}$ be a set and consider the Cartesian product $\mathbb{X}^I = (x_i)_{i \in I}$. Define

$$(x_i) \sim (x'_i) \tag{110}$$

if and only if $\mu\{i \in I \mid x_i \neq x'_i\} = 0$. Relation $\sim$ is an equivalence relation whose equivalence classes are denoted by $[x_i]$ and we define

$$^\star\mathbb{X} = \mathbb{X}^I / \sim \ . \tag{111}$$

$\mathbb{X}$ is naturally embedded into $^\star\mathbb{X}$ by mapping every $x \in \mathbb{X}$ to the constant tuple such that $x_i = x$ for every $i \in I$; we denote it by $[x]$ or, simply, by x, if no confusion can result. Any algebraic structure over $\mathbb{X}$ (group, ring, field) carries over to $^\star\mathbb{X}$ by almost pointwise extension. In particular, if $[x_i] \neq 0$, meaning that $\mu\{i \mid x_i = 0\} = 0$ we can define its inverse $[x_i]^{-1}$ by taking $y_i = x_i^{-1}$ if $x_i \neq 0$ and $y_i = 0$ otherwise. This construction yields $\mu\{i \mid y_i x_i = 1\} = 1$, whence $[y_i][x_i] = 1$ in $^\star\mathbb{X}$. The existence of an inverse for any non-zero element of a ring is indeed stated by the formula: $\forall x \, (x = 0 \vee \exists y \, (xy = 1))$.[25]

[24] Observe that, as a consequence, μ cannot be sigma-additive (in contrast to probability measures or Radon measures) in that it is *not* true that $\mu(\bigcup_n A_n) = \sum_n \mu(A_n)$ holds for an infinite denumerable sequence A_n of pairwise disjoint subsets of $\mathbb{N}$.

[25] More generally, the *Transfer Principle* states that every first-order formula is true over $^\star\mathbb{X}$ if and only if it is true over $\mathbb{X}$. Recall that a first-order formula is a formula involving quantifiers over variables but not functions.

9.3 Nonstandard reals and integers

The above general construction applies to $\mathbb{X} = \mathbb{R}$ and $I = \mathbb{N}$. The result, denoted by ${}^\star\mathbb{R}$, is a field (according to the transfer principle). By the same principle, ${}^\star\mathbb{R}$ is totally ordered by $[u_n] \leq [v_n]$ iff $\mu\{n \mid u_n > v_n\} = 0$. Call *infinitesimal* any nonstandard real number whose absolute value is smaller than any positive real number: $[\,|x_n|\,] \leq [\varepsilon]$ for any $\varepsilon \in \mathbb{R}, \varepsilon > 0$. For x and y two nonstandard real numbers, write

$$x \approx y$$

if $x - y$ is infinitesimal. Call *infinite* any nonstandard real number whose absolute value is larger than any positive real number: $[\,|x_n|\,] \geq [K]$ for any finite $K \in \mathbb{R}$. Call *finite* a nonstandard real number that is not infinite. Call *bounded* a subset $A \subseteq {}^\star\mathbb{R}$ such that there exists a finite standard positive real K such that $|x| \leq K$ for every $x \in A$.

Lemma 59 (standard part) *For any finite $[x_n] \in {}^\star\mathbb{R}$, there exists a unique standard real number $x \in \mathbb{R}$, such that $[x] - [x_n]$ is infinitesimal. We call x the* standard part *of $[x_n]$ and denote it by $\mathsf{st}([x_n])$. Infinite nonstandard reals have no standard part in $\mathbb{R}$.*

Proof To prove this, let $x = \sup\{u \in \mathbb{R} \mid [u] \leq [x_n]\}$. Since $[x_n]$ is finite, x exists and we only need to show that $[x_n] - x$ is infinitesimal. If not, then there exists $y \in \mathbb{R}, y > 0$ such that either $[x] < [x_n] - [y]$ or $[x] > [x_n] + [y]$, which both contradict the definition of x. The uniqueness of x is clear, thus we can define $\mathsf{st}([x_n]) = x$. □

It is also of interest to apply the general construction (111) to $\mathbb{X} = I = \mathbb{N}$, which results in the set ${}^\star\mathbb{N}$ of *nonstandard natural numbers*; ${}^\star\mathbb{Z}$ is defined similarly, from $\mathbb{X} = \mathbb{Z}$ and $I = \mathbb{N}$. The nonstandard set ${}^\star\mathbb{N}$ differs from $\mathbb{N}$ by the addition of *infinite natural numbers,* which are equivalence classes of sequences of integers whose essential limit is $+\infty$.

9.4 Internal functions and sets

Any sequence (f_n) of functions $f_n : \mathbb{R} \to \mathbb{R}$ pointwise defines a function $[f_n] : {}^\star\mathbb{R} \to {}^\star\mathbb{R}$ by setting

$$[f_n]([x_n]) \quad =_{\text{def}} \quad [f_n(x_n)]\,. \tag{112}$$

To justify this definition based on a representant of the equivalence class, note that $[x_n] = [x'_n]$ means that the set of integers n such that $x_n \neq x'_n$ is neglectible, and we have $f_n(x_n) = f_n(x'_n)$ for every n outside of this set. A function ${}^\star\mathbb{R} \to {}^\star\mathbb{R}$ obtained in this way is called *internal.* Properties of, and operations on, ordinary functions extend pointwise to internal functions of ${}^\star\mathbb{R} \to {}^\star\mathbb{R}$. The same notions apply to sets by considering their characteristic functions.

An internal set $A = [A_n]$ is called *hyperfinite* if $\mu\{n \mid A_n \text{ finite}\} = 1$; the *cardinal* $|A|$ of A is defined as $[\,|A_n|\,]$, where $|A_n|$ denotes the cardinal of the finite set A_n in the usual sense. For A a bounded set of hyperreals, we define its *shadow* $\mathsf{Sh}(A) \subseteq \mathbb{R}$ as follows:

$$\mathsf{Sh}(A) = \{\mathsf{st}(x) \in \mathbb{R} \mid \exists y \in A \text{ such that } y \approx x\}\,. \tag{113}$$

The *nonstandard lifting* of $f : \mathbb{R} \to \mathbb{R}$ is the internal function ${}^\star f = [f, f, f, \ldots]$. Let $f : \mathbb{R} \to \mathbb{R}$ be a standard function that is differentiable at $x \in \mathbb{R}$, and ${}^\star f = [f, f, f, \ldots]$ its nonstandard lifting. Then, we have, for every $x \in \mathbb{R}$:

$$f'(x) \quad = \quad \mathsf{st}\left(\frac{{}^\star f(y) - {}^\star f(x)}{y - x}\right) \tag{114}$$

for every $y \in {}^\star\mathbb{R}$ such that $y \approx [x]$—the proof is similar to that of Lemma 59.

Now, consider an infinite number $K \in {}^\star\mathbb{N}$ and the set

$$T = \left\{ 0, \frac{1}{K}, \frac{2}{K}, \frac{3}{K}, \ldots \frac{K-1}{K}, 1 \right\}\,. \tag{115}$$

By definition, if $K = [K_n]$, then $T = [T_n]$ with

$$T_n = \left\{ 0, \frac{1}{K_n}, \frac{2}{K_n}, \frac{3}{K_n}, \ldots \frac{K_n - 1}{K_n}, 1 \right\}$$

hence $|T| = [\,|T_n|\,] = [K_n + 1] = K + 1$. Note that the shadow of T (see (113)) is the standard interval $[0, 1]$: for any given real number a in $[0, 1]$, $\frac{\lfloor aK \rfloor}{K}$ standardizes to a.

In our nonstandard semantics, we will be using the following time basis:

$$\mathbb{T} \quad =_{\text{def}} \quad \{k\partial \mid k \in {}^\star\mathbb{N}\} \tag{116}$$

where $\partial > 0$ is some fixed infinitesimal time. This definition is reminiscent of Equation (14), in Section 2.3. The shadow of $\mathbb{T}$ is $\mathbb{R}_+$, the set of nonnegative real numbers, and that, informally speaking, $\mathbb{T}$ is both "discrete" (every element has a previous and a next element, except for 0) and "continuous" (as the shadow of $\mathbb{T}$ is $\mathbb{R}_+$).

Notations 60 Generic elements of $\mathbb{T}$ will be denoted by the special symbol $\mathfrak{t}$ or simply t when no confusion can result.

9.5 Integrals

Now, consider an internal function $f = [f_n]$ and a hyperfinite set $A = [A_n]$. The *sum* of f over A can be defined by

$$\sum_{a \in A} f(a) =_{\text{def}} \left[\sum_{a \in A_n} f_n(a) \right] .$$

If T is as above, and $f : \mathbb{R} \to \mathbb{R}$ is a standard function, we obtain

$$\sum_{t \in T} \frac{1}{|T|} {}^\star f(t) = \left[\sum_{t \in T_n} \frac{1}{|T_n|} f(t) \right] . \tag{117}$$

Now, the continuity of f implies the convergence of the Riemann sums: $\sum_{t \in T_n} \frac{1}{|T_n|} f(t) \to \int_0^1 f(t)dt$. Hence,

$$\int_0^1 f(t)dt \quad = \quad \mathsf{st}\left(\sum_{t \in T} \frac{1}{|T|} {}^\star f(t) \right). \tag{118}$$

Under the same assumptions, for any $t \in [0, 1]$,

$$\int_0^t f(u)du \quad = \quad \mathsf{st}\left(\sum_{u \in T, u \leq t} \frac{1}{|T|} {}^\star f(u) \right). \tag{119}$$

9.6 ODE

Consider the following ODE:

$$x' = f(x, t), \quad x(0) = x_0 \,. \tag{120}$$

Assume (120) possesses a solution $x : [0, 1] \to \mathbb{R}$ such that the function $t \mapsto f(x(t), t)$ is continuous. Rewriting (120) in its equivalent integral form $x(t) = x_0 + \int_0^t f(x(u), u)du$ and using (119) yields

$$x(t) = \mathsf{st}({}^\star x(t)) \text{ where } {}^\star x(t) =_{\text{def}} x_0 + \sum_{u \in T, u \leq t} \frac{1}{|T|} {}^\star f(x(u), u) \,. \tag{121}$$

One can rewrite the positive infinitesimal quantity $1/|T|$ as ∂, so that

$$T = \{\mathsf{t}_k = k\partial \mid k = 0, \ldots, |T|\}\,.$$

Then, after substituting in (121), one gets that the piecewise-constant right-continuous function ${}^\star x(\mathsf{t}), \mathsf{t}\in{}^\star\mathbb{R}, 0\leq\mathsf{t}\leq 1$ satisfies the following difference equation, for $k{=}0, \ldots, |T|{-}1$,

$$\begin{array}{lcl} {}^\star x(\mathsf{t}_{k+1}) & = & {}^\star x(\mathsf{t}_k) + \partial\,.\,{}^\star f({}^\star x(\mathsf{t}_k), \mathsf{t}_k) \\ {}^\star x(\mathsf{t}_0) & = & x_0 \end{array} \tag{122}$$

By (121), the following holds:

Theorem 61 *The solution $x(t), t \in \mathbb{R}$, of ODE (120), and ${}^\star x(\mathsf{t}), \mathsf{t} \in {}^\star\mathbb{R}_+$, the piecewise constant interpolation of the trajectory of system (122), are related by $x = \mathsf{st}({}^\star x)$.*

The same line of arguments applies for systems of ODEs possessing a unique continuous solution, i.e., x and f take their values in $\mathbb{R}^m$ in (120). Formula (122) can be seen as a *nonstandard semantics* for ODE (120). This semantics seems to depend on the choice of infinitesimal step parameter ∂. Property (121), though, expresses that

> all of these nonstandard semantics are equivalent from the standard viewpoint, regardless of the choice made for ∂. (123)

The paper [35] goes beyond our short exposure by including a direct proof of the Peano theorem on the existence of solutions of (120) when f is continuous and bounded.

9.7 Infinitesimal calculus

We conclude this background material with a summary of the section "Infinitesimal calculus" from [35]. We formulate the essential characterizations of properties of the infinitesimal calculus in nonstandard terms. f denotes a standard function and ${}^\star f = [f, f, f, \ldots]$ is its nonstandard lifting. The reader is referred to that text for the proofs.

Theorem 62

1. *The function $f : \mathbb{R} \to \mathbb{R}$ is continuous at $a \in \mathbb{R}$ if and only if ${}^\star f(x) \approx f(a)$ for all $x \approx a$.*
2. *The function $f : \mathbb{R} \to \mathbb{R}$ is uniformly continuous on set $A \subseteq \mathbb{R}$ if and only if ${}^\star f(x) \approx {}^\star f(y)$ for all $x, y \in {}^\star A$ such that $x \approx y$.*
3. *The function $f : \mathbb{R} \to \mathbb{R}$ is differentiable at $a \in \mathbb{R}$ if and only if there exists a number $b \in \mathbb{R}$ such that*
$$\frac{{}^\star f(x) - {}^\star f(a)}{x - a} \approx b\,, \text{for all } x \in {}^\star\mathbb{R}, x \approx a, x \neq a.$$
Moreover, if such a number b exists, then it equals $f'(a)$.

10 The toolkit supporting standardization

By building on the background of Section 9, we develop here the specific material we need for our developments. First, we develop the compile-time impulse analysis. Second, we formally justify our use of structural analysis in the nonstandard domain. Third, referring to Comment 4 in support of the clutch example, we formally justify our method for standardizing systems of equations. We consider systems of equations of the form

$$0 = \mathbf{H}(X', X^\bullet, V, X)\ , \tag{124}$$

where $Z =_{\text{def}} (X^\bullet, V)$ collects the dependent variables. Such systems are also written as

$$0 = H(Z, X, \partial) \tag{125}$$

after expanding X' as $\frac{X^\bullet - X}{\partial}$. In System (124), V collects the algebraic variables, X collects the state variables, and $\frac{X^\bullet - X}{\partial}$ is the nonstandard semantics of X'. Writing (124) involves both X' and $X^\bullet$ but not ∂. As (125) is obtained from (124) by performing the expansion, it involves both $X^\bullet$ and ∂ but not X'.

As an illustration, with reference to the Cup-and-Ball example, System (30) is of the form (124)—we repeat it here for convenience:

$$\left\{\begin{array}{rll} & 0 = x'' + \lambda x & (e_1)\\ & 0 = y'' + \lambda y + g & (e_2)\\ & \gamma^\bullet = [s \leq 0]; \gamma(0) = \text{F} & (k_0)\\ \texttt{if } \gamma \texttt{ then} & 0 = L^2 - (x^2 + y^2) & (k_1)\\ \texttt{and} & 0 = L^2 - (x^2 + y^2)^\bullet & (k_1^\bullet)\\ \texttt{and} & 0 = L^2 - (x^2 + y^2)^{\bullet 2} & (k_1^{\bullet 2})\\ \texttt{and} & 0 = \lambda + s & (k_2)\\ \texttt{if not } \gamma \texttt{ then} & 0 = \lambda & (k_3)\\ \texttt{and} & 0 = (L^2 - (x^2 + y^2)) - s & (k_4) \end{array}\right.$$

Its counterpart following form (125) is obtained by expanding the second derivatives by using Euler scheme:

$$\left\{\begin{array}{rll} & 0 = \frac{x^{\bullet 2} - 2x^\bullet + x}{\partial^2} + \lambda x & (e_1)\\ & 0 = \frac{y^{\bullet 2} - 2y^\bullet + y}{\partial^2} + \lambda y + g & (e_2)\\ & \gamma^\bullet = [s \leq 0]; \gamma(0) = \text{F} & (k_0)\\ \texttt{if } \gamma \texttt{ then} & 0 = L^2 - (x^2 + y^2) & (k_1)\\ \texttt{and} & 0 = L^2 - (x^2 + y^2)^\bullet & (k_1^\bullet)\\ \texttt{and} & 0 = L^2 - (x^2 + y^2)^{\bullet 2} & (k_1^{\bullet 2})\\ \texttt{and} & 0 = \lambda + s & (k_2)\\ \texttt{if not } \gamma \texttt{ then} & 0 = \lambda & (k_3)\\ \texttt{and} & 0 = (L^2 - (x^2 + y^2)) - s & (k_4) \end{array}\right.$$

Comment 63 System (124) is the generic form of systems we need to solve at Line 10 of Algorithm 13, when we eliminate higher order derivatives and shifts by introducing auxiliary variables. □

We consider the following assumptions, which hold for the nonstandard semantics of any standard mDAE system:

Assumption 6

1. *The functions* $\mathbf{H}(\cdot)$ *and* $H(\cdot)$ *are standard (they result from applying* Algorithm 13 *to our given standard mDAE model);*
2. *The state variables* X *are standard and finite (their values were assigned by the standardized dynamics of the previous mode).*

10.1 Impulse Analysis

We refer the reader to Section 2.5 about the standardization of the clutch model at mode changes, and particularly System (25) in it. As a prerequisite to standardization, we had to identify which variables could become impulsive at mode changes (the two torques in this example), and then, we had to eliminate them.

To identify impulsive and non-impulsive variables in a systematic way, we develop the *impulse analysis,* which consists in abstracting hyperreals with their "magnitude order" compared to the infinitesimal ∂. Since our original DAE models are all standard, only the occurrence of the infinitesimal ∂ will have a role in this matter. The impulse analysis will be useful as a preparation step for standardization.

Definition 64 (impulse order and analysis)

1. *Given a system of equations such as* (124) *or* (125)*, say that a dependent variable x has* impulse order *(or simply* order*) $\mathfrak{o} \in \mathbb{R}$, if the solution of the considered system is such that $x\partial^{\mathfrak{o}}$ is provably a finite non-zero (standard) real number. We denote by $[\![x]\!]$ the impulse order of x. By convention, the constant 0 has impulse order $-\infty$.*
2. *Say that x is* impulsive *if $[\![x]\!] > 0$.*
3. *The* impulse analysis *of a system of equations such as* (124) *or* (125) *is the system of constraints satisfied by the impulse orders of the dependent variables of the system.*

10.1.1 The rules of impulse analysis

Figures 21 and 22 display the rules defining the translation of a system of equations of the form (124) or (125) into its impulse analysis, for the restricted class where only rational expressions are involved.

$$\begin{array}{rcl} e & ::= & 0 \mid c \mid \partial \mid x \mid e^c \mid e+e \mid e \times e \\ E & ::= & e = e \mid E \texttt{ and } E \end{array}$$

Figure 21: Syntax: E is a system of one or several equations $e = e$. An expression e is 0, a nonzero (standard) real constant c, the infinitesimal ∂, a variable x, the monomial e^c, a sum, or a product.

$$\begin{array}{lrcl} \text{(R1)} & [\![0]\!] & = & -\infty \\ \text{(R2)} & [\![c]\!] & = & 0 \\ \text{(R3)} & [\![\partial]\!] & = & -1 \\ \text{(R4)} & [\![e^c]\!] & = & c[\![e]\!] \\ \text{(R5)} & [\![e_1 \times e_2]\!] & = & [\![e_1]\!] + [\![e_2]\!] \\ \text{(R6)} & [\![e_1 + e_2]\!] & \leq & \max\{[\![e_1]\!], [\![e_2]\!]\} \end{array}$$

$$\frac{E \vdash e = e'}{[\![E]\!] \vdash [\![e]\!] = [\![e']\!]} \quad \text{(R7)}$$

$$\frac{\left.\begin{array}{l} E \vdash x = y + e \quad \text{or} \\ E \vdash 0 = y - x + e \end{array}\right\} \text{ and } \quad E \nvdash y = x - e}{E \vdash E \texttt{ and } y = x - e} \quad \text{(R8)}$$

Figure 22: Rules: The left column displays the impulse order of the primitive expressions. Rule (R7) indicates that $[\![e]\!] = [\![e']\!]$ is an equation of the impulse analysis $[\![E]\!]$ if $e = e'$ is an equation of E; Rule (R8) indicates that, if E involves the equation $x = y + e$ but not the equation $y = x - e$, then we augment E with the latter, i.e., we saturate E with the rule $x = y + e \implies y = x - e$.

Figure 21 describes the syntax of a mini-language specifying such systems of equations. The left column of Figure 22 gives the rules for mapping expressions to their corresponding impulse orders. All the rules are self-explanatory, with the exception of Rule (R6), where the inequality deserves some explanation. The sum e_1+e_2, the dominant terms in the expansion of the e_i's as power series over ∂ may compensate each other for their impulse orders. For an example of this, see equation (e_1^{∂}) in System (25): rewriting this equation as $a_1(\omega_1) = \frac{\omega_1^{\bullet} - \omega_1}{\partial} - b_1(\omega_1)\tau_1$, we see a case of strict inequality for (R6) since $a_1(\omega_1)$ has order zero, whereas it is equal to the difference of two terms of order one, see Section 10.1.4 for details.

We will use Rule (R6) in the following way, thereby reinforcing it. Consider an equation

$$e : z = x + y\,.$$

We can rewrite e in the following equivalent ways: $0 = x + y - z$, $x = z - y$, or $y = z - x$. To each of them we apply the max rule. This yields, for the impulse analysis of equation e, the following system of constraints:

$$\text{impulse analysis of } e : \left\{ \begin{array}{lcl} [\![z]\!] \leq \max\{[\![x]\!], [\![y]\!]\} & ; & [\![0]\!] \leq \max\{[\![x]\!], [\![y]\!], [\![z]\!]\} \\ [\![x]\!] \leq \max\{[\![z]\!], [\![y]\!]\} & ; & [\![y]\!] \leq \max\{[\![x]\!], [\![z]\!]\} \end{array} \right. \tag{126}$$

Note that the constraint $[\![0]\!] \leq \ldots$ is vacuously satisfied since $[\![0]\!] = -\infty$. Also note that, among the three nontrivial inequalities of (126), at least two must be saturated. We will use impulse analysis (126) for handling sums of terms. This reinforcement of the max rule is formalized by Rule (R8) of Figure 22, which mechanizes the association, to equation e of (126), of its different rewritings.

Using the rules of Figures 21 and 22 in the numerical expressions, we map any system of rational equations of the form (124) or (125) into a system of constraints over impulse orders.

To cover functions beyond polynomials, we need to extend $\mathbb{R} \cup \{-\infty\}$ with $+\infty$. In this extension, we take the convention that $-\infty + \infty = -\infty$, justified by the equality $0 \times x = 0$ for any nonstandard x. For functions $f(x) = \sum_{k=0}^{\infty} a_k x^k$ that can be represented as absolutely converging power series, we then get

$$[\![f(x)]\!] = [\![\sum_{k=0}^{\infty} a_k x^k]\!] = [\![x]\!].\sup(A), \text{ where } A = \{k \mid a_k \neq 0\} \tag{127}$$

is the support of the series and $\sup(A)$ is the supremum of set A. In particular, if $[\![x]\!] > 0$ and if the support of the series is infinite, we get $[\![f(x)]\!] = +\infty$.

10.1.2 Impulse analysis of systems of equations for restarts

Here we particularize the impulse analysis to systems of equations of the form (124), where the only reason for ∂ to occur is the expansion of derivatives using the Euler scheme:

$$0 = \mathbf{H}\left(\frac{X^\bullet - X}{\partial}, X^\bullet, V, X\right)$$

The dependent variables are $X^\bullet, V$. It will be convenient to introduce the auxiliary variables

$$U =_{\text{def}} X^\bullet - X \,,$$

so that the systems we consider take the following form:

$$\text{dependent variables } X^\bullet, V, U \text{ in } : \left\{ \begin{array}{lcl} 0 & = & \mathbf{H}\left(\frac{U}{\partial}, X^\bullet, V, X\right) \\ U & = & X^\bullet - X \end{array} \right. \tag{128}$$

The following condition for System (128) can be assumed, based on physical considerations (restart values for an ODE or a DAE cannot be impulsive in a physically meaningful model):

Assumption 7 *Since X is a state, both X (a known value) and $X^\bullet$ must be finite.*

First, the impulse orders $[\![X]\!]$ are all known, from previous nonstandard instants. Next, from Assumption 7 we deduce the inequalities

$$[\![X^\bullet]\!] \leq 0 \quad \text{and} \quad [\![U]\!] \leq 0 \,. \tag{129}$$

The impulse orders $[\![V]\!]$ are a priori unknown. We have, however, more prior information, thanks to the structural analysis. As part of both the simpler algorithm ExecRun (Algorithm 10) and revisited algorithm ExecRun (Algorithm 13), the auxiliary algorithm SolveConflict (Algorithm 9) is called. At the considered mode change, we thus know which consistency equation(s) of the new mode was/were conflicting with the dynamics of the previous mode. Formally, call G=0 the subsystem collecting

all the equations that were erased by algorithm SolveConflict at Line 6 of Algorithm 10 or Line 7 of Algorithm 13. As a result, $G{=}0$ is no longer satisfied at the considered mode change, and thus, G defines a tuple R of finite nonzero variables called *residuals*, by setting

$$R = G\,. \tag{130}$$

An example of residual in the clutch is $r = \omega_1 - \omega_2$, which is both finite and nonzero at mode change $\gamma : \text{F} \to \text{T}$; this information was found by the structural analysis. Finally, the system of equations that we need to solve collects all the above items, namely:

$$\text{dependent variables } X^\bullet, V, U, R \text{ in} \quad : \quad \left\{ \begin{array}{rcl} 0 & = & \mathbf{H}\left(\frac{U}{\partial}, X^\bullet, V, X\right) \\ U & = & X^\bullet - X \\ R & = & G \end{array} \right. \tag{131}$$

$$\text{prior information on impulse orders} \quad : \quad \left\{ \begin{array}{rcl} [\![\frac{1}{\partial}]\!] & = & 1 \\ [\![X^\bullet]\!] & \leq & 0 \\ [\![U]\!] & \leq & 0 \\ [\![R]\!] & = & 0 \end{array} \right. \tag{132}$$

Decomposing algebraic dependent variables into impulsive and non-impulsive ones In the following, we assume that the vector functions $\mathbf{H}$ or H can be represented as absolutely convergent power series in their arguments. We can then apply the rules of Figures 21 and 22, complemented with (127), to derive the system of equations that impulse orders must satisfy. In particular, this allows us to partition the set V of algebraic variables as

$$V \quad = \quad W^{\mathsf{i}} \cup W \tag{133}$$

where W^{i} collects the *impulsive variables*, having impulse order > 0, and W collects the *non-impulsive variables*. In Section 11.3, we will propose a numerical scheme for computing restarts that only requires the splitting of algebraic dependent variables into impulsive and non-impulsive ones. For now, let us develop the impulse analysis for the two transitions $\gamma : \text{T} \to \text{F}$ and $\gamma : \text{F} \to \text{T}$ of System (16) assuming linear f_i as in Eq. (24).

10.1.3 Example: mode change $\gamma : \text{T} \to \text{F}$ of the clutch

We develop this example here based on the rules of Figures 21 and 22. To better illustrate this mechanical reasoning, we forbid ourselves algebraic manipulations that otherwise could help simplifying a manual analysis. We repeat here the corresponding system of equations with its expansion (128)—there is no residual for this mode change:

$$\left\{ \begin{array}{ll} \frac{u_1}{\partial} = a_1(\omega_1) + b_1(\omega_1)\tau_1 & (e_1^\partial) \\ \frac{u_2}{\partial} = a_2(\omega_2) + b_2(\omega_2)\tau_2 & (e_2^\partial) \\ \tau_1 = 0 & (e_5) \\ \tau_2 = 0 & (e_6) \\ u_1 = \omega_1^\bullet - \omega_1 & \\ u_2 = \omega_2^\bullet - \omega_2 & \end{array} \right. \tag{134}$$

We then saturate System (134) using Rule (R8) of Figure 22 (added equations are in blue):

$$\left\{ \begin{array}{l} \frac{u_1}{\partial} = a_1(\omega_1) + b_1(\omega_1)\tau_1 \;;\; a_1(\omega_1) = \frac{u_1}{\partial} - b_1(\omega_1)\tau_1 \;;\; b_1(\omega_1)\tau_1 = \frac{u_1}{\partial} - a_1(\omega_1) \\ \frac{u_2}{\partial} = a_2(\omega_2) + b_2(\omega_2)\tau_2 \;;\; a_2(\omega_2) = \frac{u_2}{\partial} - b_2(\omega_2)\tau_2 \;;\; b_2(\omega_2)\tau_2 = \frac{u_2}{\partial} - a_2(\omega_2) \\ \tau_1 = 0 \\ \tau_2 = 0 \\ u_1 = \omega_1^\bullet - \omega_1 \;;\; u_1 + \omega_1 = \omega_1^\bullet \;;\; u_1 - \omega_1^\bullet = -\omega_1 \\ u_2 = \omega_2^\bullet - \omega_2 \;;\; u_2 + \omega_2 = \omega_2^\bullet \;;\; u_2 - \omega_2^\bullet = -\omega_2 \end{array} \right. \tag{135}$$

and we consider the prior information (132):

$$[\![\omega_i^\bullet]\!] \leq 0 \; ; \; [\![u_i]\!] \leq 0 \; ; \; [\![\frac{1}{\partial}]\!] = 1 \tag{136}$$

We now apply the rules of Figure 22 to System (135), then use (136):

$$\begin{cases} 1 + [\![u_1]\!] \leq 0 \; ; \; 0 \leq 1 + [\![u_1]\!] \; ; \; \text{vacuous} \\ 1 + [\![u_2]\!] \leq 0 \; ; \; 0 \leq 1 + [\![u_2]\!] \; ; \; \text{vacuous} \\ [\![\tau_1]\!] = -\infty \\ [\![\tau_2]\!] = -\infty \\ [\![u_1]\!] \leq 0 \; ; \; \text{vacuous} \; ; \; \text{vacuous} \\ [\![u_2]\!] \leq 0 \; ; \; \text{vacuous} \; ; \; \text{vacuous} \end{cases} \tag{137}$$

where "vacuous" indicates that the equation sitting in the corresponding position in System (135) generates a vacuously satisfied constraint. System (137) shows that $[\![u_i]\!] = -1$, meaning that u_i is continuous. Note that System (137) uniquely determines the impulse orders for all dependent variables.

10.1.4 Example: mode change $\gamma : \text{F} \to \text{T}$ of the clutch

We repeat here the corresponding system of equations with its expansion (128), and we include the residual $r =_{\text{def}} \omega_1 - \omega_2$ (from structural analysis, we know it is nonzero), highlighted in red:

$$\begin{cases} \frac{u_1}{\partial} = a_1(\omega_1) + b_1(\omega_1)\tau_1 & (e_1^\partial) \\ \frac{u_2}{\partial} = a_2(\omega_2) + b_2(\omega_2)\tau_2 & (e_2^\partial) \\ \omega_1^\bullet - \omega_2^\bullet = 0 & (e_3^\bullet) \\ \tau_1 + \tau_2 = 0 & (e_4) \\ u_1 = \omega_1^\bullet - \omega_1 \\ u_2 = \omega_2^\bullet - \omega_2 \\ r = \omega_1 - \omega_2 \end{cases} \tag{138}$$

We then saturate System (138) using Rule (R8) of Figure 22 (added equations are in blue):

$$\begin{cases} \frac{u_1}{\partial} = a_1(\omega_1) + b_1(\omega_1)\tau_1 \; ; \; a_1(\omega_1) = \frac{u_1}{\partial} - b_1(\omega_1)\tau_1 \; ; \; b_1(\omega_1)\tau_1 = \frac{u_1}{\partial} - a_1(\omega_1) \\ \frac{u_2}{\partial} = a_2(\omega_2) + b_2(\omega_2)\tau_2 \; ; \; a_2(\omega_2) = \frac{u_2}{\partial} - b_2(\omega_2)\tau_2 \; ; \; b_2(\omega_2)\tau_2 = \frac{u_2}{\partial} - a_2(\omega_2) \\ \omega_1^\bullet - \omega_2^\bullet = 0 \; ; \; \omega_1^\bullet = \omega_2^\bullet \\ \tau_1 + \tau_2 = 0 \; \; ; \; \tau_1 = -\tau_2 \\ u_1 = \omega_1^\bullet - \omega_1 \; ; \; u_1 + \omega_1 = \omega_1^\bullet \; ; \; u_1 - \omega_1^\bullet = -\omega_1 \\ u_2 = \omega_2^\bullet - \omega_2 \; ; \; u_2 + \omega_2 = \omega_2^\bullet \; ; \; u_2 - \omega_2^\bullet = -\omega_2 \\ r = \omega_1 - \omega_2 \end{cases} \tag{139}$$

We also consider the prior information (132):

$$[\![\omega_i^\bullet]\!] \leq 0 \; ; \; [\![u_i]\!] \leq 0 \; ; \; [\![r]\!] = 0 \; ; \; [\![\frac{1}{\partial}]\!] = 1 \; . \tag{140}$$

We now apply the rules of Figure 22 to System (139), then use (140):

$$\begin{cases} 1 \leq [\![\tau_1]\!] \; ; \; \text{vacuous} \; ; \; [\![\tau_1]\!] \leq 1 \\ 1 \leq [\![\tau_2]\!] \; ; \; \text{vacuous} \; ; \; [\![\tau_2]\!] \leq 1 \\ \text{vacuous} \; ; \; [\![\omega_1^\bullet]\!] = [\![\omega_2^\bullet]\!] \\ \text{vacuous} \; ; \; [\![\tau_1]\!] = [\![\tau_2]\!] \\ \text{vacuous} \; ; \; \text{vacuous} \; ; \; 0 \leq \max\{[\![u_1]\!], [\![\omega_1^\bullet]\!]\} \\ \text{vacuous} \; ; \; \text{vacuous} \; ; \; 0 \leq \max\{[\![u_2]\!], [\![\omega_2^\bullet]\!]\} \\ 0 = [\![r]\!] = [\![\omega_1]\!] = [\![\omega_2]\!] \\ [\![\omega_i^\bullet]\!] \leq 0 \; , \; [\![u_i]\!] \leq 0 \; , \; i = 1, 2 \end{cases} \tag{141}$$

which proves $[\![\tau_1]\!] = [\![\tau_2]\!] = 1$, expressing that the two torques are impulsive. Note, on the other hand, that (141) does not fully determine the impulse orders of $\omega_i^\bullet$ and u_i, since one may, e.g., have $[\![\omega_i^\bullet]\!] < 0$ and $[\![u_i]\!] = 0$.

10.1.5 Using impulse analysis in code generation

Code generation for restarts consists in standardizing System (124) or System (125). Recall that standardizing systems of equations requires more care than standardizing numbers, due to impulsive behaviors and singularity issues that result.

We can exploit the impulse analysis through three different approaches, which we illustrate by using the clutch example and particularly the mode change $\gamma : \text{F} \to \text{T}$ when the clutch gets engaged. The system of equations for the corresponding restart is (25), which we recall for convenience:

$$\begin{cases} \omega_1^\bullet = \omega_1 + \partial.(a_1(\omega_1) + b_1(\omega_1)\tau_1) & (e_1^\partial) \\ \omega_2^\bullet = \omega_2 + \partial.(a_2(\omega_2) + b_2(\omega_2)\tau_2) & (e_2^\partial) \\ \omega_1^\bullet - \omega_2^\bullet = 0 & (e_3^\bullet) \\ \tau_1 + \tau_2 = 0 & (e_4) \end{cases}$$

As we have shown, the two torques τ_1 and τ_2 are impulsive of order 1.

Eliminating impulsive variables When this is practical, the simplest method is to eliminate impulsive variables from the restart system, namely the two torques in System (25). The details were given in Section 2.5.2 and the resulting reduced system of equations is

$$\omega_1^\bullet = \omega_2^\bullet = \frac{b_2(\omega_2)\omega_1 + b_1(\omega_1)\omega_2}{b_1(\omega_1) + b_2(\omega_2)} + \partial \frac{a_1(\omega_1)b_2(\omega_2) + a_2(\omega_2)b_1(\omega_1)}{b_1(\omega_1) + b_2(\omega_2)}, \tag{142}$$

whose standardization is simply achieved by substituting $\partial \leftarrow 0$. No numerical information is provided on the torques with this method.

This is a satisfactory solution when elimination of impulsive variables is practical. In our example, they entered linearly in the restart system, so that elimination was straightforward. When this is not the case, elimination becomes costly or even impossible. We thus need to look for alternatives.

Rescaling impulsive variables Since $[\![\tau_i]\!] = 1$, we apply the change of variable $\tau_i = \partial\hat{\tau}_i$, which transforms System (138) into:

$$\begin{cases} \omega_1^\bullet - \omega_1 = a_1(\omega_1) + b_1(\omega_1)\hat{\tau}_1 & (e_1^\partial) \\ \omega_2^\bullet - \omega_2 = a_2(\omega_2) + b_2(\omega_2)\hat{\tau}_2 & (e_2^\partial) \\ \omega_1^\bullet - \omega_2^\bullet = 0 & (e_3^\bullet) \\ \hat{\tau}_1 + \hat{\tau}_2 = 0 & (e_4) \end{cases} \tag{143}$$

which yields again the solution (142) for $\omega_1^\bullet = \omega_2^\bullet$. However, the rescaled value $\hat{\tau}_1$ can now be computed. Rescaling impulsive variables is simpler than eliminating them. Unfortunately, this method does not work in full generality since impulse orders can be infinite as we have shown in (127). The last method addresses such cases, at the price of a possibly poor numerical conditioning.

Bruteforce solving of the restart system Consider once more System (25), replace in it the infinitesimal time step ∂ by a small real positive time step $\delta > 0$, and expand the derivatives as before. This yields

$$\begin{cases} \omega_1^\bullet = \omega_1 + \delta.(a_1(\omega_1) + b_1(\omega_1)\tau_1) & (e_1^\delta) \\ \omega_2^\bullet = \omega_2 + \delta.(a_2(\omega_2) + b_2(\omega_2)\tau_2) & (e_2^\delta) \\ \omega_1^\bullet - \omega_2^\bullet = 0 & (e_3^\bullet) \\ \tau_1 + \tau_2 = 0 & (e_4) \end{cases} \tag{144}$$

Then, it will be proved in Theorem 91 of Section 11.3 that *solving System* (144) *for its dependent variables and then discarding the values found for the impulsive variables yields a converging approximation for the states* ω_1^+ *and* ω_2^+ *at restart.* Of course, numerical conditioning for System (144) is likely to be less good than for System (143), so that rescaling is recommended when impulse orders are finite.

Qualitative information obtained from impulse analysis The dynamical profile of impulsive variables yields interesting information about the nature of the impulsive behaviors. For the clutch example, we get the following profile for $[\![\tau_i]\!] : \ldots, 0, 1, 0, \ldots$, where the 1 occurs at the event $\gamma : \text{F} \to \text{T}$. This indicates a Dirac-like impulse. As a second example (not developed here), the impulse analysis of the Cup-and-Ball example with inelastic impact yields the following profile for the impulse order of the tension $[\![\lambda]\!] : \ldots, 0, 1, 0, \ldots$, where the 1 occurs at the event following the mode change $\gamma : \text{F} \to \text{T}$, i.e., when System (34) gets executed to evaluate restart values for velocities. This again indicates a Dirac impulse for the tension. Note that having a profile $\ldots, 0, k, 0, \ldots$ for an impulsive variable, where k is a positive integer, still indicates a Dirac behavior. The amplitude does not matter: only the profile matters. Developing a systematic analysis of this kind requires further study and is left for future investigations.

10.2 Nonstandard Structural Analysis

The motivation for this section was developed in Section 2.5. The following two questions will be addressed:

1. We apply structural analysis to the nonstandard semantics. Is this justified? In the standard setting, the structural analysis was motivated by the quest for low cost criteria for almost everywhere nonsingularity of systems of algebraic equations. Can we have a similar argument here?

2. We need to formalize and generalize the Comment 4 of Section 2.5.2, where we discussed the difference between standardizing functions and standardizing equations.

We begin with question 1. Algorithm 13 returns at its Line 10 systems of algebraic equations of the form $H(Z, X, \partial) = 0$ introduced in (125), where tuple Z collects the dependent variables and tuple X collects the state variables. We justify the call of the atomic action $\mathsf{Eval}(\sigma, H)$ by the fact that "H is structurally nonsingular". But the justification of structural analysis was developed in Section 6.1.1, i.e., in a standard setting. We thus need a justification for taking the liberty of using it for nonstandard semantics as well. Formally, we consider nonstandard systems of equations of the form

$$\begin{array}{l} {}^{\star}H\,(Z, {}^{\star}X, \partial) = 0, \text{ where } {}^{\star}X \in {}^{\star}\mathbb{R}^m, \; Z \text{ collects the dependent (nonstandard) variables, } {}^{\star}H = [H, H, H, \ldots] \text{ is the nonstandard lifting of the standard function } H, \text{ and } \partial = [\delta_n] \text{ is infinitesimal.} \end{array} \tag{145}$$

Definition 65 *Say that* ${}^{\star}H\,(Z, {}^{\star}X, \partial) = 0$ *is* structurally nonsingular *if and only if so is the standard equation* $H\,(Z, X, \delta) = 0$ *in the sense of Section* 6.1.1*, where* $\delta > 0$ *is a standard step size.*

Lemma 66 *Lemma* 13 *extends to nonstandard systems of the form* (145)*.*

Proof Select ${}^{\star}z = [z_n]$ and ${}^{\star}x = [x_n]$ satisfying ${}^{\star}H\,({}^{\star}z, {}^{\star}x, \partial) = 0$. By definition, we have $[H\,(z_n, x_n, \delta_n)] \sim [0, 0, \ldots]$, where relation $\sim$ was defined in (110). Equivalently,

$$\mu(A) = 1, \text{ where } A = \{n \mid H\,(z_n, x_n, \delta_n) = 0\} \tag{146}$$

and μ is the finitely additive measure over $\mathbb{N}$ introduced in (109). Now, assume that ${}^{\star}H$ is structurally nonsingular according to Definition 65. Then, using notations (55) of Section 6.1.1, for every $n \in A$, there exist $\mathbf{U}_n =_{\text{def}} U_{H_n}(z_n, x_n, \delta_n)$ and $\mathbf{V}_n =_{\text{def}} V_{H_n}(z_n, x_n, \delta_n)$ satisfying property (54) with respect to the system $H\,(Z_n, X_n, \delta_n) = 0$.

Define the internal sets $\mathbf{U} = [\mathbf{U}_n]$ and $\mathbf{V} = [\mathbf{V}_n]$. We have ${}^{\star}\lambda(\mathbf{V} \setminus \mathbf{U}) = 0$, where ${}^{\star}\lambda$ is the internal set function ${}^{\star}\lambda = [\lambda, \lambda, \lambda, \ldots]$. For every $n \in A$, every $\mathcal{J}_n \in \mathbf{V}_n$ yields an invertible matrix, implying that $\mathcal{J} =_{\text{def}} [\mathcal{J}_n]$ also yields an invertible matrix. As a result, we have exhibited two sets

U and **V** satisfying property (54) with respect to the system ${}^\star H\,(Z, {}^\star X, \partial) = 0$. □

Definition 65 thus formally justifies our shameless use of the structural analysis in nonstandard domains. Note that Definition 65 tells nothing about how **U** is—it could be infinitely small. Neither does it require uniqueness of the solutions of the considered equations.

We now move to question 2 and investigate how equations should be standardized.

Theorem 67 (standardizing equations) *Consider a square system of the form (145), and assume that H is of class $\mathcal{C}^1$. The following two properties hold:*

1. *If the standard pair (z, x) satisfies $H(z, x, 0)=0$ and the Jacobian $\mathbf{J}_Z H$ at the point $(z, x, 0)$ is nonsingular, then, for every nonstandard pair $({}^\star x, \partial)$ such that $({}^\star x, \partial) \approx (x, 0)$, there exists ${}^\star z$ such that $\mathsf{st}({}^\star z)=z$ and ${}^\star H\,({}^\star z, {}^\star x, \partial) =0$.*
2. *Vice-versa, let $({}^\star z, {}^\star x, \partial)$ be a finite nonstandard triple satisfying ${}^\star H\,({}^\star z, {}^\star x, \partial) = 0$, and let $(z, x, 0) = \mathsf{st}({}^\star z, {}^\star x, \partial)$ be its standard part. Then $H(z, x, 0) = 0$ holds.*

Under these assumptions, the standardization of the equation ${}^\star H\,({}^\star z, {}^\star x, \partial) =0$ is independent from the particular value for ∂ we have chosen (provided that it is infinitesimal).

Corollary 68 *We can standardize ${}^\star H(Z, X, \partial)=0$ as the system $H(Z, X, 0)=0$, provided that the latter is structurally nonsingular.*

The nonsingularity condition is needed for statement 1 to hold. We proceed to the proof of Theorem 67.

Proof We successively prove the two statements. For Statement 1, by the Implicit Function Theorem applied to the standard function H, there exists a $\mathcal{C}^1$-class function $(x', \delta) \mapsto G(x', \delta)$ such that $H(G(x', \delta), x', \delta)=0$ holds for (x', δ) ranging over a neighborhood V of $(x, 0)$. Let $({}^\star x, \partial) = ([x_n], [\delta_n])$ be such that $\mathsf{st}({}^\star x, \partial)=(x, 0)$. Then, by definition of the standard part, there exists $B \subseteq \mathbb{N}$ with $\mu(B) = 1$ such that, for every $n \in B$, $(x_n, \delta_n) \in V$. The sequence $(G(x_n, \delta_n))_{n\in B}$ satisfies

$$H(G(x_n, \delta_n), x_n, \delta_n)=0, \text{ for every } n \in B. \tag{147}$$

We extend the sequence $(G(x_n, \delta_n))_{n\in B}$ to $\mathbb{N}$ by assigning arbitrary values to indices $n \in \mathbb{N} \setminus B$ and consider the nonstandard ${}^\star z =_{\text{def}} [G(x_n, \delta_n)]$. By (147), we get ${}^\star H\,({}^\star z, {}^\star x, \partial) =0$.

For Statement 2, let the finite triple $({}^\star z, {}^\star x, \partial)$ satisfy ${}^\star H\,({}^\star z, {}^\star x, \partial) = 0$ and let $(z, x, 0)=\mathsf{st}({}^\star z, {}^\star x, \partial)$. Since $(z, x, 0)\approx({}^\star z, {}^\star x, \partial)$ and H is of class $\mathcal{C}^1$, we can apply Statement 2 of Theorem 62 to derive $H(z, x, 0) \approx {}^\star H\,({}^\star z, {}^\star x, \partial) = 0$, whence $H(z, x, 0) = 0$ follows since $H(z, x, 0)$ is standard. □

10.3 DAE of index zero

Invoking the implicit function theorem, one can also address DAEs of index 0, i.e., of the form:

$$F(X', X) = 0 \tag{148}$$

where the Jacobian $\nabla_V F(V, X)$ is invertible in a neighborhood of the solution of the equation $F(V, X) = 0$, where V is the vector of dependent variables.

Theorem 69 *Let $F(X', X) = 0$ be a structurally nonsingular DAE system of index 0, and let ${}^\star F = [F, F, F, \dots]$ be the nonstandard lifting of F. Consider the following nonstandard transition system having $X^\bullet$ as dependent variables:*

$$0 = {}^\star F\left(\frac{X^\bullet - X}{\partial}, X\right), X(0) = X_0 \tag{149}$$

Then, system (149) possesses a unique solution and we denote by ${}^\star X(\mathsf{t}), \mathsf{t}\in{}^\star\mathbb{R}_+$ its piecewise constant interpolation. Furthermore, the solution X of DAE system $F(X', X)=0, X(0)=X_0$ satisfies $X = \mathsf{st}({}^\star X)$.

Proof Recall that the time basis of nonstandard semantics is $\mathbb{T} = \{k{\times}\partial \mid k{\in}{}^\star\mathbb{N}\}$. Denoting by t_k the generic element of $\mathbb{T}$, transition system (149) rewrites:

$$ {}^\star F\left(\frac{X(\mathsf{t}_{k+1}) - X(\mathsf{t}_k)}{\partial}\,,\, X(\mathsf{t}_k)\right) = 0\,. \tag{150}$$

Using the Implicit Function Theorem, let G be the standard function such that $X' = G(X)$ solves (148) for X', and set ${}^\star G = [G, G, G, \dots]$. Then, (150) rewrites

$$ X(\mathsf{t}_{k+1}) = X(\mathsf{t}_k) + \partial.{}^\star G(X(\mathsf{t}_k)),\ X(0) = X_0\,. \tag{151}$$

Theorem 69 follows from Theorem 61 applied to (151). □

In the next two sections we investigate the standardization of the blocks returned by Algorithm 13, as identified in Section 7.3.5. Since, in this section, we pay attention to the status standard/nonstandard, we will make this status explicit in our development whenever needed.

10.4 Long modes of mDAE

We use the notations and results of Section 7.3.5. More precisely, we reuse notations (103) with $\prime$ replacing $\bullet$, and we invoke Lemma 42. We consider a system $B_\prime(H, N) = (B^{\mathsf{t}}_\prime(H, N), B^{\mathsf{h}}_\prime(H, N))$ returned by Algorithm 13 for long modes. By construction of Algorithm 13, the algebraic system $G_\Sigma = 0$, where $G_\Sigma =_{\text{def}} B^{\mathsf{t}}_\prime(H, N)$, is structurally nonsingular and admits $\bar{G}_\Sigma = 0$, where $\bar{G}_\Sigma =_{\text{def}} B^{\mathsf{h}}_\prime(H, N)$, as associated consistency conditions.

At this point, we need to relate the above blocks to their standard continuous-time counterparts. The reduced index systems $(G_\Sigma, \bar{G}_\Sigma)$ returned by Algorithm 8 (IndexReduc) in case of success, coincide with the two systems returned by the Σ-method, when replacing differentiation by shifting (see the emphasized sentence when introducing this algorithm). In our study of standardization, we will also need to consider the original outcome of the Σ-method (using differentiation); we denote this pair by $(G^c_\Sigma, \bar{G}^c_\Sigma)$, where the superscript ^{c} refers to continuous time. Continuous-time systems $(G^c_\Sigma, \bar{G}^c_\Sigma)$ define the leading equations and consistency conditions of a (standard) DAE system that is generically regular. In the following theorem, we denote by $X = (X(t))_{t\in\mathbb{R}_+}$ the solution of this DAE system.

Since we are concerned with standardization, we carefully distinguish between H, a standard vector function, and ${}^\star H = [H, H, \dots]$, its nonstandard lifting.

Theorem 70 *The dAE system $B^{\mathsf{t}}_\bullet({}^\star H, N){=}0$, with consistency conditions $B^{\mathsf{h}}_\bullet({}^\star H, N){=}0$, possesses a unique solution. Its piecewise constant interpolation, denoted by ${}^\star X(\mathsf{t}), \mathsf{t}{\in}{}^\star\mathbb{R}_+$, satisfies $X = \mathsf{st}({}^\star X)$. Note that this standardization is independent from the particular infinitesimal time step ∂.*

Theorem 70 formalizes the practical informal rule that

> for long modes, the standardization of $B^{\mathsf{t}}_\bullet(H, N){=}0$ with consistency conditions $B^{\mathsf{h}}_\bullet(H, N){=}0$ is $B^{\mathsf{t}}_\prime(H, N){=}0$ with consistency conditions $B^{\mathsf{h}}_\prime(H, N){=}0$, (152)

where

$$\begin{aligned} B^{\mathsf{h}}_\prime(f, \prime n) &=_{\text{def}} \begin{bmatrix} f \\ f' \\ \vdots \\ f^{\prime(n-1)} \end{bmatrix} \text{ and} \\ B^{\mathsf{t}}_\prime(f, \prime n) &=_{\text{def}} f'^n\ , \end{aligned} \tag{153}$$

whereas $B^{\mathsf{t}}_\prime(H, N)$ and $B^{\mathsf{h}}_\prime(H, N)$ are constructed from (153) as in Notations 41.

Proof The theorem is proved by componentwise induction on N, with the help of the Lemma 71 to follow. □

Lemma 71 *Consider a structurally nonsingular nonstandard dAE system of the form*

$$\begin{aligned} {}^\star H(X^\bullet, Y^\bullet, X, Y) &=_{\text{def}} \begin{bmatrix} {}^\star G(X^\bullet, Y^\bullet, X, Y) \\ {}^\star g(X) \\ {}^\star g(X^\bullet) \end{bmatrix} \\ &= 0 \end{aligned} \tag{154}$$

with leading variables $X^\bullet, Y^\bullet$, and consider

$$\begin{aligned} {}^\star \widehat{H}(X^\bullet, Y^\bullet, X, Y) &=_{\text{def}} \begin{bmatrix} {}^\star G(X^\bullet, Y^\bullet, X, Y) \\ {}^\star g(X) \\ {}^\star(\mathbf{J}g)(X).X' \end{bmatrix} \\ &= 0 \end{aligned} \tag{155}$$

where $\mathbf{J}g(X)$ is the Jacobian of the function g at X. Let $({}^\star X(\mathfrak{t}), {}^\star Y(\mathfrak{t})), \mathfrak{t} \in {}^\star\mathbb{R}_+$, and $({}^\star \widehat{X}(\mathfrak{t}), {}^\star \widehat{Y}(\mathfrak{t})), \mathfrak{t} \in {}^\star\mathbb{R}_+$, be the piecewise constant interpolations of the solutions of nonstandard dAE systems (154) and (155), respectively. Assume that we know a standard DAE block

$$\widehat{G}(X', Y', X, Y) = 0$$

such that the solution $(X(t), Y(t))_{t\in\mathbb{R}}$ of

$$\begin{aligned} \widehat{H}(X', Y', X, Y) &=_{\text{def}} \begin{bmatrix} \widehat{G}(X', Y', X, Y) \\ g(X) \\ \mathbf{J}g(X).X' \end{bmatrix} \\ &= 0 \end{aligned} \tag{156}$$

exists, is unique, and satisfies $(X, Y) = \mathsf{st}\left({}^\star \widehat{X}, {}^\star \widehat{Y}\right)$. Then, we also have $(X, Y) = \mathsf{st}({}^\star X, {}^\star Y)$.

Proof By statement 3 of Theorem 62, we have

$${}^\star g(X^\bullet) \quad = \quad {}^\star g(X) + \partial.{}^\star(\mathbf{J}g)(X).\left(X' + \varepsilon\right)$$

where ε is infinitesimal. Hence ${}^\star H$ rewrites

$${}^\star H(X^\bullet, Y^\bullet, X, Y) \quad = \quad \begin{bmatrix} {}^\star G(X^\bullet, Y^\bullet, X, Y) \\ {}^\star g(X) \\ {}^\star g(X) + \partial.{}^\star(\mathbf{J}g)(X).(X' + \varepsilon) \end{bmatrix}$$

Since ${}^\star H(X^\bullet, Y^\bullet, X, Y) = 0$ implies in particular ${}^\star g(X) = 0$, (154) rewrites

$$\begin{bmatrix} {}^\star G(X^\bullet, Y^\bullet, X, Y) \\ {}^\star g(X) \\ {}^\star(\mathbf{J}g)(X).(X' + \varepsilon) \end{bmatrix} = 0$$

and removing the infinitesimal ε does not change the standardization of this dAE system. We conclude by invoking the assumption of the lemma. □

10.5 Mode changes of mDAE

Assumption 6 (stated on page 73) is in force throughout this section. We study the standardization of mode change events. Having successfully performed the nonstandard structural analysis, the system at mode changes can be put in the generic form (124), and we use the decomposition (133) of the algebraic variables into impulsive and non-impulsive ones. For convenience, we repeat these formulas here:

$$(124) : X' = \tfrac{X^\bullet - X}{\partial} \text{ in } 0 = \mathbf{H}(X', X^\bullet, V, X) \qquad (133) : \quad V = W^{\mathsf{i}} \cup W$$

Note that, in (124), the infinitesimal parameter ∂ is involved in both time (by being the time step of $\mathbb{T}$) and space (since derivatives x' in the original standard model are expanded as $\frac{x^\bullet - x}{\partial}$). Concrete instances of this are found in the clutch example, e.g., in System (16).

Compared with the previous section that dealt with long modes, the novelty, here, is that we take a different target domain for the standardization, namely the domain of *discrete-time* transition systems—standard transition systems indexed by $\mathbb{N}$. Clearly, we cannot expect mapping to this domain the entire trajectory of System (124). We only aim at standardizing the finite prefix of it needed to specify the restart conditions for the new mode. The "$\bullet$" shift operator defines the discrete time of this dynamics, hence we do not need to eliminate ∂ from the time domain. In contrast, we will have to eliminate the occurrences of the infinitesimal parameter ∂ acting in space.

Consider again the clutch example and in particular System (25), used to determine the restart values $\omega_1^\bullet, \omega_2^\bullet$ and the current torques τ_1, τ_2 from the current values for the states ω_1, ω_2. The occurrence of ∂ in the right-hand sides of equations (e_1^∂), (e_2^∂) is the difficulty, since ∂ is nonstandard—see Comment 4 regarding this.

The standardization of (124) relies on Theorem 67. Unfortunately, it is not true in general that substituting 0 for ∂ in (124) yields a structurally nonsingular system with respect to the dependent variables $Z =_{\text{def}} (X^\bullet, V)$. We thus need further work to be able to invoke Theorem 67. This will consist in eliminating, from System (124), the impulsive variables. For the example of the clutch, the corresponding analysis was developed in Section 2.5.2. In the following, we propose a systematic approach.

To this end, we also use the rewriting of (124), i.e., we use both:

$$\begin{aligned} (124) \quad &: \quad 0 = \mathbf{H}(X', X^\bullet, V, X) \\ (125) \quad &: \quad 0 = H(Z, X, \partial) \end{aligned}$$

where we recall that the form (125) is obtained by expanding X' as $\frac{X^\bullet - X}{\partial}$ in (124)—the same notational convention is used in the assumption below. Using decomposition (133) of algebraic variables into impulsive and non-impulsive ones: $V = W^{\mathsf{i}} \cup W$, and writing

$$Y =_{\text{def}} (X^\bullet, W) \ ,$$

we assume the following about (124):

Assumption 8 *There exists a reduced system* $\mathbf{K}(X', X^\bullet, W, X) = 0$ *with dependent variables* Y*, satisfying the following two conditions:*

1. *System* $K(Y, X, 0){=}0$ *is structurally nonsingular, where* $K(Y, X, \partial)$ *is derived from* $\mathbf{K}(X', X^\bullet, W, X)$ *as explained above;*

2. *The considered model is a reduction of the original one, in that, equivalently:*

$$\begin{aligned} \mathbf{K}(X', X^\bullet, W, X) = 0 \quad &\Longleftrightarrow \quad \exists W^{\mathsf{i}}. \mathbf{H}(X', X^\bullet, V, X) = 0 \\ K(Y, X, \partial) = 0 \quad &\Longleftrightarrow \quad \exists W^{\mathsf{i}}. H(Z, X, \partial) = 0 \end{aligned}$$

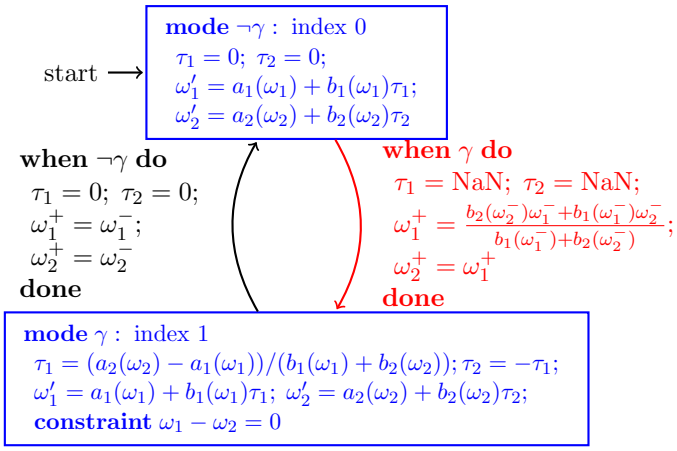


Figure 23: Standardization of ExecRun for the ideal clutch (Figure 20). Blocks have been standardized and then symbolically pivoted. x^- is the previous value of state variable x, i.e., the left limit of x when exiting a mode; x^+ is the value for restart. Long modes are colored blue; non-impulsive (resp. impulsive) state-jumps are colored black (resp. red). The dynamics in mode $\neg\gamma$ is defined by an ODE system, while in mode γ, it is defined by an over-determined index-1 DAE system consisting of an ODE system coupled to an algebraic constraint. In the transition from mode $\neg\gamma$ to mode γ, variables τ_1 and τ_2 are impulsive, and their standardization is undefined. This explains why they are set to NaN (Not a Number).

In words, we can eliminate, from the original model, the impulsive variables (belonging to W^{i}), and the result is structurally nonsingular for $\partial = 0$. We constructed such a reduced system in Section 2.5.2 by eliminating the torques for the released-to-engaged mode change in the clutch example.

Theorem 72 *Assumptions 6 and 8 are in force. Then, standardizing System (124) is performed by solving, for Y, the structurally nonsingular system $K(Y, X, 0) = 0$. Note that this standardization is independent from the particular infinitesimal time step ∂.*

Note that the latter system is standard.

Proof Theorem 72 is a consequence of Theorem 67. □

Theorem 72 allows us to compute the restart values for the state variables that are not impulsive, which is sufficient to restart the dynamics in the coming long mode (impulsive variables at mode change cannot be initial conditions for state variables in the new mode).

We have shown in Section 2.5.2 that Assumption 8 holds for the clutch example. In general, the method would consist in, first, computing the decomposition (133), and, then, eliminating impulsive variables. Figure 23 shows the effective code resulting from the standardization of the labeled automaton of Figure 20.

To get practical, Assumption 8 relies on computer algebraic techniques for eliminating variables from systems of equations. Algebraic elimination is easy for systems in which impulsive variables enter linearly. If they enter polynomially, Gröbner bases can be used, but at a high computational cost. Effective elimination techniques may not exist in general, or may be too costly. Hence, the alternative based on the numerical scheme developed in Section 11.3 is of great practical interest.

11 Main results

This section collects results that are essential in grounding our approach. First, we prove that our approach is intrinsic, in that the actual code we generate does not depend on the particular

nonstandard scheme used, instead of the explicit first-order Euler scheme, to expand the derivative. Then, we compare our approach with known schemes, for subclasses for which they are known to exist. Finally, we propose a numerical scheme to compute restart values at mode changes, in full generality, for programs accepted by our structural analysis. This numerical scheme does not require eliminating impulsive variables using computer algebra.

11.1 What if we change the interpretation of the derivative?

So far, an essential step in our reasoning was the interpretation (74) for the derivative ω'. However, several nonstandard interpretations of the derivative can be equally considered. What happens if we take an interpretation different from (74)? We first discuss this on the clutch example, then provide a general result.

11.1.1 The clutch example

If the system is within a long continuous mode, the following alternative expansion exists for the derivative of ω: for $n \in \mathbb{N}$ finite but otherwise arbitrary, $\omega' \approx \frac{(\omega^\bullet - \omega)^{\bullet n}}{\partial}$. More generally, for $\mathbf{a}=(\alpha_P, \ldots, \alpha_N)$ such that $P \leq N$, $N \geq 0$, $\alpha_N \neq 0$, and $\sum_{n=P}^{N} \alpha_n = 1$, one has:

$$\omega' \approx \frac{1}{\partial} \sum_{n=P}^{N} \alpha_n (\omega^\bullet - \omega)^{\bullet n} \ . \tag{157}$$

Let us investigate how the analysis of the clutch is modified when using the nonstandard interpretation

$$\omega'_{\mathbf{a}} =_{\text{def}} \frac{1}{\partial} \sum_{n=P}^{N} \alpha_n (\omega^\bullet - \omega)^{\bullet n} \tag{158}$$

for the derivative. The nonstandard semantics of the clutch model is:

$$\left\{ \begin{array}{rll} & & \omega'_{\mathbf{a}1} = f_1(\omega_1, \tau_1) \quad (e_1^\partial) \\ & & \omega'_{\mathbf{a}2} = f_2(\omega_2, \tau_2) \quad (e_2^\partial) \\ \texttt{if } \gamma & \texttt{then} & \omega_1 - \omega_2 = 0 \quad (e_3) \\ & \texttt{and} & \tau_1 + \tau_2 = 0 \quad (e_4) \\ \texttt{if not } \gamma & \texttt{then} & \tau_1 = 0 \quad (e_5) \\ & \texttt{and} & \tau_2 = 0 \quad (e_6) \end{array} \right. \tag{159}$$

Algorithm 13 tells us that we must shift forward (e_3) by $N+1$ and keep other equations unchanged; other latent equations $(e_3^{\bullet n})$ for $n \leq N$ are erased when solving the conflict with the previous mode:

$$\left\{ \begin{array}{rll} & & \omega'_{\mathbf{a}1} = f_1(\omega_1, \tau_1) \quad (e_1^\partial) \\ & & \omega'_{\mathbf{a}2} = f_2(\omega_2, \tau_2) \quad (e_2^\partial) \\ \texttt{if } \gamma & \texttt{then} & \omega_1^{\bullet(N+1)} - \omega_2^{\bullet(N+1)} = 0 \quad (e_3^{\bullet(N+1)}) \\ & \texttt{and} & \tau_1 + \tau_2 = 0 \quad (e_4) \\ \texttt{if not } \gamma & \texttt{then} & \tau_1 = 0 \quad (e_5) \\ & \texttt{and} & \tau_2 = 0 \quad (e_6) \end{array} \right. \tag{160}$$

Using $\tau_1 = -\tau_2 =_{\text{def}} \tau$, the resetting equations when entering the mode $\gamma = \text{T}$ are

$$\left\{ \begin{array}{l} \omega'_{\mathbf{a}1} = f_1(\omega_1, \tau) \\ \omega'_{\mathbf{a}2} = f_2(\omega_2, -\tau) \\ \omega_1^{\bullet(N+1)} - \omega_2^{\bullet(N+1)} = 0 \end{array} \right. \tag{161}$$

Assuming again that the f_i's are linear in τ, i.e., of the form $f_i(\omega_i,\tau) = a_i(\omega_i) + b_i(\omega_i)\tau$, (161) rewrites:

$$\begin{cases} \sum_{n=P}^{N} \alpha_n(\omega_1^\bullet - \omega_1)^{\bullet n} = \partial.(a_1(\omega_1) + b_1(\omega_1)\tau) \\ \sum_{n=P}^{N} \alpha_n(\omega_2^\bullet - \omega_2)^{\bullet n} = \partial.(a_2(\omega_2) - b_2(\omega_2)\tau) \\ \omega_1^{\bullet(N+1)} - \omega_2^{\bullet(N+1)} = 0 \end{cases} .$$

We can eliminate τ:

$$\begin{array}{rcl} 0 & = & \sum_{n=P}^{N} \alpha_n\big(b_2(\omega_2)(\omega_1^\bullet - \omega_1)^{\bullet n} + b_1(\omega_1)(\omega_2^\bullet - \omega_2)^{\bullet n}\big) \\ & & -\partial.(a_1(\omega_1)b_2(\omega_2) + a_2(\omega_2)b_1(\omega_1)) \\ 0 & = & \omega_1^{\bullet(N+1)} - \omega_2^{\bullet(N+1)} \end{array} \quad (162)$$

We can zero the term in ∂ in the first equation: doing so keeps the system structurally nonsingular, hence it is safe. It then remains to standardize

$$\begin{array}{rcl} 0 & = & \sum_{n=P}^{N} \alpha_n\big(b_2(\omega_2)(\omega_1^\bullet - \omega_1)^{\bullet n} + b_1(\omega_1)(\omega_2^\bullet - \omega_2)^{\bullet n}\big) \\ 0 & = & \omega_1^{\bullet(N+1)} - \omega_2^{\bullet(N+1)} \end{array} \quad (163)$$

and, finally, solve it for the dependent variable $\omega^{\bullet(N+1)} =_{\text{def}} \omega_1^{\bullet(N+1)} = \omega_2^{\bullet(N+1)}$. For $n = P,\ldots,N$, $\omega_i^{\bullet n}$ standardizes as the left-limit ω_i^- for $i = 1,2$, whereas $\omega^{\bullet(N+1)}$ standardizes as the right-limit ω^+. Hence, (163) standardizes as

$$0 = \alpha_N\big(b_2(\omega_2^-)(\omega^+ - \omega_1^-) + b_1(\omega_1^-)(\omega^+ - \omega_2^-)\big)$$

which does not depend on α_N since $\alpha_N \neq 0$. Finally, we get

$$\omega^+ = \frac{b_2(\omega_2^-)\omega_1^- + b_1(\omega_1^-)\omega_2^-}{b_1(\omega_1^-) + b_2(\omega_2^-)} ,$$

which coincides with (27).

11.1.2 A general result

In this section, we consider again the generic nonstandard interpretation (158) for the derivative. We investigate under which assumptions the result of our whole compilation suite is independent from the particular way derivatives are expanded when mapping the considered mDAE system to its nonstandard semantics. We consider the following assumption on S, thus characterizing a restricted class of ∂AE:

Assumption 9 *mDAE-system S involves only long modes, no transient modes.*

Let us compare, under Assumption 9 and Comment 63, the results of Algorithm 10 on S for the following two nonstandard semantics for the derivative:

$$\begin{array}{rcl} \partial.x' & = & x^\bullet - x \\ x^\bullet & = & \partial.x' + x \end{array} \quad (164)$$

and

$$\begin{array}{rcl} \partial.x'_{\mathbf{a}} & = & \sum_{n=P}^{N} \alpha_n(x^\bullet - x)^{\bullet n} \\ \alpha_N.x^{\bullet(N+1)} & = & \partial.x'_{\mathbf{a}} + \sum_{n=P}^{N}(\alpha_n - \alpha_{n-1})x^{\bullet n} \end{array} \quad (165)$$

where $\mathbf{a} = (\alpha_P,\ldots,\alpha_N)$ satisfies $P{\leq}N$, $N{\geq}0$, $\alpha_N{\neq}0$, and $\sum_{n=P}^{N}\alpha_n = 1$, whereas $\alpha_{P-1}{=}0$ by convention. Note that nothing prevents P from being negative. For the two formulas, we have given both the direct and inverse forms. Equations (164) are obtained by taking $P{=}N{=}0$ and $\alpha_N{=}1$ in (165).

Comparing the structural analyses

Let S be an mDAE and let H_S be the system H returned by Algorithm 10 at Line 6, i.e., the outcome of index reduction, for both a long mode and a mode change event.

We will consider the following problem: how should H_S be modified to become the outcome of the index reduction of S, when derivatives are expanded using (165) instead of the first-order Euler expansion (164)? We denote by ${}^\star S_{\mathbf{a}}$ and ${}^\star S$ the corresponding two nonstandard semantics for S. Regarding this question, two lines of Algorithm 10 are important: Line 3 and Line 6. We investigate them successively.

Line 3 of Algorithm 10 This consists in applying the Σ-method to the original mDAE system S, within a given mode; let $G{=}0$ be the dynamics of S in this mode. The outcome of this algorithm only depends on the weighted bipartite graph $\mathcal{G}$ associated to G. In the original Σ-method, the weight of an edge (f, x) of graph $\mathcal{G}$ is equal to the highest differentiation degree of x in f. Similarly, if G is a dAE system (in discrete time), the weight of an edge (f, x) of $\mathcal{G}$ is equal to the highest shifting degree of x in f. Having recalled this, we can now compare the two expansions:

- *Using expansion* (164)*:*
  1. We begin with DAE system G and its bipartite graph $\mathcal{G}$;
  2. Replacing DAE system G by dAE system ${}^\star G$ using expansion (164) yields ${}^\star\mathcal{G}$, that is identical to $\mathcal{G}$.
- *Using expansion* (165)*:*
  1. We begin with DAE system G and its bipartite graph $\mathcal{G}$;
  2. Replacing DAE system G by dAE system ${}^\star G$ using expansion (165) yields ${}^\star\mathcal{G}_{\mathbf{a}}$, in which the vertices and edges are exactly those in $\mathcal{G}$ and all weights have been multiplied by

$$M =_{\text{def}} N{+}1 \ .$$

The case $N = 0$ is the base case of the forward first-order Euler scheme, so that we do not need to consider it. Therefore, in the sequel, we assume $N > 0$, i.e., $M > 1$.

Using the notations of (65), let $(c_f)_{f\in F}$ and $(c^{\mathbf{a}}_f)_{f\in F}$ be the offsets associated to ${}^\star\mathcal{G}$ and ${}^\star\mathcal{G}_{\mathbf{a}}$. Then, by Lemma 25, one gets:

Lemma 73 *The Σ-method succeeds on ${}^\star\mathcal{G}$ if and only if it succeeds on ${}^\star\mathcal{G}_{\mathbf{a}}$, and, in this case, $c^{\mathbf{a}}_f = M \times c_f$ for every $f{\in}F$ and $d^{\mathbf{a}}_x = M \times d_x$ for every $x{\in}X(F)$.*

In particular, each guarded equation in ${}^\star S_{\mathbf{a}}$ corresponds to an equation in ${}^\star S$ shifted M times. We denote by

$$(G_\Sigma, \overline{G}_\Sigma) \text{ and } (G^{\mathbf{a}}_\Sigma, \overline{G}^{\mathbf{a}}_\Sigma) \tag{166}$$

the corresponding two returns of the Σ-method.

Line 6 of Algorithm 10 If the current instant sits within a long mode, then this line just erases the consistency conditions $\overline{G}_\Sigma$ and $\overline{G}^{\mathbf{a}}_\Sigma$. So, we are left with G_Σ if we use expansion (164), and $G^{\mathbf{a}}_\Sigma$ if we use expansion (165). Now, dividing the infinitesimal step ∂ by M in G_Σ yields $G^{\mathbf{a}}_\Sigma$. Theorem 70 tells us that the standardizations of the two systems G_Σ and $G^{\mathbf{a}}_\Sigma$ yield the same DAE system.

As such, the difficulty is the handling of mode changes by Line 6, when the two different expansions are used. Is the handling of conflicts preserved, when the two different expansions are used? More precisely, for $\mathfrak{t}_o \in \mathbb{T}$ an instant of mode change:

- Let $G{=}0$ and ${}^{\bullet}G{=}0$ be the DAE system at the current ($\mathfrak{t} \geq \mathfrak{t}_o$) and previous ($\mathfrak{t} < \mathfrak{t}_o$) modes, respectively; we denote by (c_f, d_x) and (c_g^-, d_y^-) the offsets returned by the Σ-method for the current and previous mode, respectively;
- Let $(G_\Sigma, \overline{G}_\Sigma)$ and $(G_\Sigma^{\mathbf{a}}, \overline{G}_\Sigma^{\mathbf{a}})$ be the returns of the Σ-method when applied to $G{=}0$ and $G^{\mathbf{a}}{=}0$;
- Let $({}^{\bullet}G_\Sigma, {}^{\bullet}\overline{G}_\Sigma)$ and $({}^{\bullet}G_\Sigma^{\mathbf{a}}, {}^{\bullet}\overline{G}_\Sigma^{\mathbf{a}})$ be the returns of the Σ-method when applied to ${}^{\bullet}G{=}0$ and ${}^{\bullet}G^{\mathbf{a}}{=}0$;
- We denote by Δ and $\Delta^{\mathbf{a}}$, and by Φ and $\Phi^{\mathbf{a}}$, the contexts and sets of facts constructed by ExecRun if expansions (164) and (165) are used.

Line 6 takes as inputs the systems $(G_\Sigma \cup \overline{G}_\Sigma) \setminus \Phi$ and $(G_\Sigma^{\mathbf{a}} \cup \overline{G}_\Sigma^{\mathbf{a}}) \setminus \Phi^{\mathbf{a}}$. When applying SolveConflict, the dependent variables for $(G_\Sigma \cup \overline{G}_\Sigma) \setminus \Phi$ are all the variables of this system, both leading and non-leading, whose values were not already set by executing the previous nonstandard instant. The same holds for the dependent variables of $(G_\Sigma^{\mathbf{a}} \cup \overline{G}_\Sigma^{\mathbf{a}}) \setminus \Phi^{\mathbf{a}}$. The following result holds:

Theorem 74 *Set* $F =_{\text{def}} (G_\Sigma \cup \overline{G}_\Sigma) \setminus \Phi$ *with dependent variables defined by context* Δ, *and* $F^{\mathbf{a}} =_{\text{def}} (G_\Sigma^{\mathbf{a}} \cup \overline{G}_\Sigma^{\mathbf{a}}) \setminus \Phi^{\mathbf{a}}$ *with dependent variables defined by context* $\Delta^{\mathbf{a}}$. *Then, the mappings*

$$\begin{array}{lll} \psi_f : F^{\mathbf{a}} \mapsto F & \text{defined by} & \psi_f\left(f^{{}^{\bullet M(c_f - m)}}\right) = f^{{}^{\bullet(c_f-m)}} \\ \psi_x : X(F^{\mathbf{a}}) \mapsto X(F) & \text{defined by} & \psi_x\left(x^{{}^{\bullet M(d_x - m)}}\right) = x^{{}^{\bullet(d_x-m)}} \end{array}$$

preserve the Dulmage-Mendelsohn decompositions of $\mathcal{G}_F$ *and* $\mathcal{G}_{F^{\mathbf{a}}}$, *i.e.,* $g \in \beta^{\mathbf{a}}_{\mathfrak{D}/\mathfrak{E}/\mathfrak{U}}$ *if and only if* $\psi_f(g) \in \beta_{\mathfrak{D}/\mathfrak{E}/\mathfrak{U}}$, *and* $y \in \beta^{\mathbf{a}}_{\mathfrak{D}/\mathfrak{E}/\mathfrak{U}}$ *if and only if* $\psi_x(y) \in \beta_{\mathfrak{D}/\mathfrak{E}/\mathfrak{U}}$.

The proof of this theorem decomposes into a series of lemmas. Let $f{=}0$ range over the set of equations of system $G{=}0$, and let $n = c_f$ be the equation offset computed for f when applying the Σ-method to $G{=}0$.

Lemma 75

1. *The following formulas hold:*

$$\begin{array}{rcl@{\quad}l} G_\Sigma \cup \overline{G}_\Sigma = \bigcup_{f \in G} B(f, c_f) & \text{and} & G_\Sigma^{\mathbf{a}} \cup \overline{G}_\Sigma^{\mathbf{a}} = \bigcup_{f \in G} B(f, Mc_f) & (a) \\ \Delta = \bigcup_{f \in {}^{\bullet}G} B(f, c_f^- - 1) & \text{and} & \Delta^{\mathbf{a}} = \bigcup_{f \in {}^{\bullet}G} B(f, M\,c_f^- - 1) & (b) \\ \Phi = \bigcup_{f \in {}^{\bullet}G \cap G} B(f, k_f) & \text{and} & \Phi^{\mathbf{a}} = \bigcup_{f \in {}^{\bullet}G \cap G} B(f, K_f) & (c) \\ \text{where } k_f = \min(c_f^- - 1, c_f) & \text{and} & K_f = \min(M\,c_f^- - 1, Mc_f) & (d) \end{array} \tag{167}$$

2. *If* $c_f^- \leq c_f$, *then* $k_f = c_f^- - 1$ *and* $K_f = M\,c_f^- - 1$;
   If $c_f^- > c_f$, *then* $k_f = c_f$ *and* $K_f = Mc_f$.

Proof For the first statement, (167-*a*) is a direct consequence of Lemma 73 and Lemma 42 from Section 7.3.5. (167-*b*) and (167-*c*) follow from Definition 40 and Lemma 73. For the second statement, we note that, since $M > 1$, $c_f^- \leq c_f$ is equivalent to $c_f^- - \frac{1}{M} \leq c_f$, or $M\,c_f^- - 1 \leq Mc_f$. The case $c_f^- \leq c_f$ follows. The other case is proved similarly. □

In the following lemma, we use Notations 41 and recall that $X(G)$ denotes the set of variables involved in the set G of functions. We also introduce the following notations.

Notations 76 For G=0 a DAE system, we extend by convention its offsets $(c_f)_{f\in G}$ and $(d_x)_{x\in X(G)}$ to arbitrary f and x by setting $c_f = d_x = -\infty$ if $f \notin G$ and $x \notin X(G)$. For $-\infty \leq m$ and $0 \leq n$, and ζ a variable or a function, we set

$$\mathbf{B}(\zeta, m, n) =_{\text{def}} \begin{bmatrix} \zeta^{\bullet \widehat{m}} \\ \zeta^{\bullet(\widehat{m}+1)} \\ \vdots \\ \zeta^{\bullet n} \end{bmatrix}, \text{ where } \widehat{m} = \max(m, 0)$$

with the convention that $\mathbf{B}(\zeta, m, n)$ is empty if $m > n$.

Lemma 77 *Set* $F =_{\text{def}} (G_\Sigma \cup \overline{G}_\Sigma) \setminus \Phi$ *and* $F^{\mathbf{a}} =_{\text{def}} (G^{\mathbf{a}}_\Sigma \cup \overline{G}^{\mathbf{a}}_\Sigma) \setminus \Phi^{\mathbf{a}}$. *Then:*

1. *F and $F^{\mathbf{a}}$ rewrite as follows:*

$$F = \bigcup_{f\in G} \mathbf{B}(f, k_f{+}1, c_f) \text{ and } F^{\mathbf{a}} = \bigcup_{f\in G} \mathbf{B}(f, K_f{+}1, Mc_f) \ . \tag{168}$$

2. *Let Z and $Z^{\mathbf{a}}$ be the sets of dependent variables of F=0 and $F^{\mathbf{a}}$=0. Then:*

$$Z = \bigcup_{x\in X(G)} \mathbf{B}(x, d_x^-, d_x) \text{ and } Z^{\mathbf{a}} = \bigcup_{x\in X(G)} \mathbf{B}(x, Md_x^-, Md_x) \ . \tag{169}$$

Proof Obvious. □

If $c_f^- \leq c_f$, then $k_f{+}1 = c_f^-$ and $K_f{+}1 = Mc_f^-$; thus, $\mathbf{B}(f, k_f{+}1, c_f)$ and $\mathbf{B}(f, K_f{+}1, c_f)$ are both nonempty. If $c_f^- > c_f$, then $k_f{+}1 = c_f{+}1$ and $K_f{+}1 = Mc_f{+}1$; thus, $\mathbf{B}(f, k_f{+}1, c_f)$ and $\mathbf{B}(f, K_f{+}1, c_f)$ are both empty. Hence, (168) rewrites:

$$F = \bigcup_{f\in G: c_f^- \leq c_f} \mathbf{B}(f, c_f^-, c_f) \text{ and } F^{\mathbf{a}} = \bigcup_{f\in G: c_f^- \leq c_f} \mathbf{B}(f, Mc_f^-, Mc_f) \ . \tag{170}$$

Similarly, (169) rewrites:

$$Z = \bigcup_{x\in X(G): d_x^- \leq d_x} \mathbf{B}(x, d_x^-, d_x) \text{ and } Z^{\mathbf{a}} = \bigcup_{x\in X(G): d_x^- \leq d_x} \mathbf{B}(x, Md_x^-, Md_x) \ . \tag{171}$$

In words, $F^{\mathbf{a}}$ and its set of dependent variables are obtained, from F and its set of dependent variables, via an M-stretching. From formulas (170) and (171), we deduce the following result:

Lemma 78 *Let $f \in G$ and $x \in X(G)$. Then, $(f^{\bullet k}, x^{\bullet l})$ is an edge of $\mathcal{G}_F$ if and only if $(f^{\bullet Mk}, x^{\bullet Ml})$ is an edge of $\mathcal{G}_{F^{\mathbf{a}}}$.*

The situation is illustrated in Figures 24, 25, and 26, for the clutch and two variants of it, for $N = 1$ and $N = 2$.

Procedure for constructing matchings of maximal cardinality for F and $F^{\mathbf{a}}$

We want to design a procedure for determining such matchings so that the two constructions parallel each other. This will allow us to compare the results.

Step 1: starting from G_Σ and $G^{\mathbf{a}}_\Sigma$ By construction, G_Σ and $G^{\mathbf{a}}_\Sigma$ are both structurally nonsingular and determine the leading variables of the new mode, given a consistent valuation of other variables. They both admit a complete matching.

$$\begin{array}{l} \text{Let } \mathcal{N}_1 \text{ be a complete matching for } G_\Sigma. \text{ Its edges have the form} \\ \left(f^{\bullet c_f}, x^{\bullet d_x}\right) \text{ for appropriate pairs } (f, x) \in G \times X(G). \text{ Then,} \\ \mathcal{N}_1^{\mathbf{a}} =_{\text{def}} \left\{ \left(f^{\bullet Mc_f}, x^{\bullet Md_x}\right) \,\middle|\, \left(f^{\bullet c_f}, x^{\bullet d_x}\right) \in \mathcal{N}_1 \right\} \text{ is a complete} \\ \text{matching for } G^{\mathbf{a}}_\Sigma. \end{array} \tag{172}$$

This parallel construction is illustrated in Figure 24.

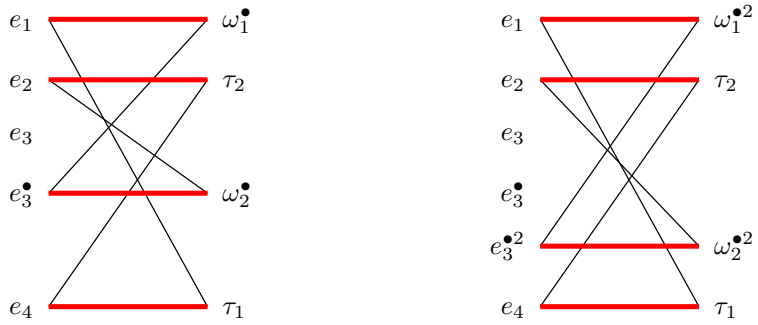


Figure 24: The clutch is an example in which $c_f^- = c_f$ holds. Maximal cardinality matchings for $N = 1$ (left) and $N = 2$ (right) are given. Subsequently applying the Dulmage-Mendelsohn decomposition returns $\beta_{\mathfrak{D}} = \{e_3\}$ on the left, and $\beta_{\mathfrak{D}} = \{e_3, e_3^\bullet\}$ on the right.

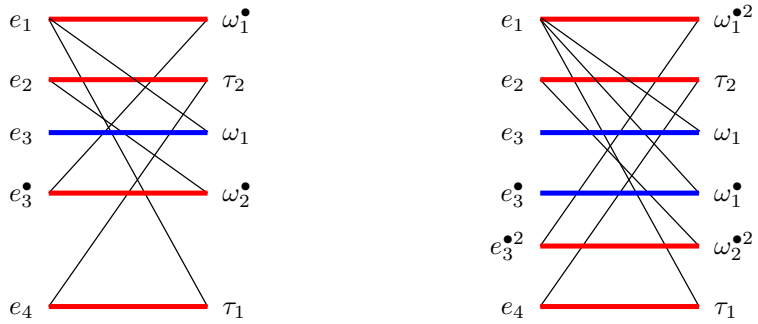


Figure 25: An example with $c_f^- < c_f$: a variation on the clutch. The previous mode does not involve $\omega_1^\bullet$, hence ω_1 remains a dependent variable when entering the new mode. Maximal cardinality matchings for $N = 1$ (left) and $N = 2$ (right) are shown. With reference with the example of Figure 24, the added edges correspond to consistency equations and are colored in blue. Subsequently applying the Dulmage-Mendelsohn decomposition returns $\beta_{\mathfrak{D}} = \emptyset$ on both. Actually, the two matchings are complete. As a result, the mode change is fully determined, despite the new mode for the clutch having underdetermined consistency conditions (because the common rotation velocity is free).

Step 2: Removing, from the set of dependent variables of G_Σ and $G_\Sigma^{\mathbf{a}}$, the variables involved in the contexts Δ and $\Delta^{\mathbf{a}}$ Let

$$Y =_{\text{def}} \left\{ x \in X(G) \;\middle|\; x^{\bullet d_x} \in X(\Delta) \right\} \text{ and } Y^{\mathbf{a}} =_{\text{def}} \left\{ x \in X(G) \;\middle|\; x^{\bullet M d_x} \in X(\Delta^{\mathbf{a}}) \right\} .$$

Then, $Y = Y^{\mathbf{a}}$ holds. Denote by $\widehat{G}_2$ (resp. $\widehat{G}_2^{\mathbf{a}}$) the system G_Σ (resp. $G_\Sigma^{\mathbf{a}}$) where the variables involved in the context Δ (resp. $\Delta^{\mathbf{a}}$) have been removed. The bipartite graphs of $\widehat{G}_2$ and $\widehat{G}_2^{\mathbf{a}}$, denoted by $\widehat{\mathcal{G}}_2$ and $\widehat{\mathcal{G}}_2^{\mathbf{a}}$, are isomorphic. Hence:

> If $\widehat{\mathcal{N}}_2$ denotes a matching of maximal cardinality of $\widehat{\mathcal{G}}_2$, then, $\widehat{\mathcal{N}}_2^{\mathbf{a}} =_{\text{def}} \left\{ \left(f^{\bullet M c_f}, x^{\bullet M d_x} \right) \;\middle|\; \left(f^{\bullet c_f}, x^{\bullet d_x} \right) \in \widehat{\mathcal{N}}_2 \right\}$ is a matching of maximal cardinality for $\widehat{\mathcal{G}}_2^{\mathbf{a}}$. (173)

In particular, their Dulmage-Mendelsohn decompositions are isomorphic too. An example of this is illustrated in Figure 26.

We start from $\widehat{\mathcal{N}}_2$ and $\widehat{\mathcal{N}}_2^{\mathbf{a}}$, perform their respective Dulmage-Mendelsohn decompositions, and remove, from both bipartite graphs, their respective overdetermined blocks $\beta_{\mathfrak{D}}$ and $\beta_{\mathfrak{D}}^{\mathbf{a}}$. The resulting bipartite graphs are denoted by $\mathcal{G}_2$ and $\mathcal{G}_2^{\mathbf{a}}$: they are also isomorphic. Hence:

> Let $\mathcal{N}_2$ be a matching of maximal cardinality for $\mathcal{G}_2$. The formula $\mathcal{N}_2^{\mathbf{a}} =_{\text{def}} \left\{ \left(f^{\bullet M c_f}, x^{\bullet M d_x} \right) \;\middle|\; \left(f^{\bullet c_f}, x^{\bullet d_x} \right) \in \mathcal{N}_2 \right\}$ defines a matching of maximal cardinality for $\mathcal{G}_2^{\mathbf{a}}$. (174)

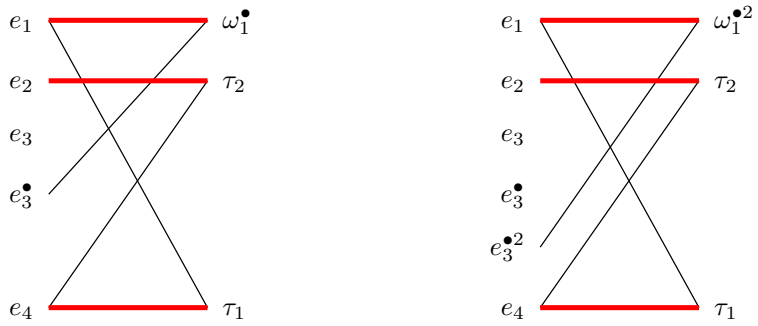


Figure 26: An example with $c_f^- > c_f$: another variation on the clutch. The previous mode determines $\omega_2^{\bullet 2}$, hence $\omega_2^\bullet$ is no longer a dependent variable when entering the new mode. Maximal cardinality matchings for $N = 1$ (left) and $N = 2$ (right) are shown. Subsequently applying the Dulmage-Mendelsohn decomposition on the left-hand graph returns $\beta_{\mathfrak{D}} = \mathcal{G}_F$, hence Line 6 of Algorithm 10 returns $H = \emptyset$. No equation of the new mode is satisfied at the first instant and we move to the next instant. The same holds for the right-hand graph.

$\mathcal{N}_2$ involves all equations of $\mathcal{G}_2$ since, by Lemma 17, both $\mathcal{G}_2$ and $\mathcal{G}_2^{\mathbf{a}}$ have empty overdetermined blocks in their respective Dulmage-Mendelsohn decompositions.

Step 3: Adding to G_Σ the consistency subsystem $\overline{G}_\Sigma$ and subtracting Φ; performing the corresponding changes to $G_\Sigma^{\mathbf{a}}$ We start from $\mathcal{G}_2, \mathcal{N}_2$ and $\mathcal{G}_2^{\mathbf{a}}, \mathcal{N}_2^{\mathbf{a}}$. We do not need to remove equations belonging to Φ and $\Phi^{\mathbf{a}}$, since such equations, if any, have already been removed at Step 2, when moving from $\widehat{\mathcal{N}}_2$ to $\mathcal{N}_2$. Therefore, the only final modification we need to perform is adding the consistency equations. The following result holds:

Lemma 79 *1. Let $f \in G$ such that $c_f^- \leq c_f$ and let $(f^{\bullet c_f}, x^{\bullet d_x}) \in \mathcal{N}_2$. For every m such that $0 < m \leq c_f - c_f^-$, the edge $(f^{\bullet(c_f - m)}, x^{\bullet(d_x - m)})$ is an augmenting path of $\mathcal{N}_2$ with respect to system F.*

2. Let $f \in G$ such that $c_f^- \leq c_f$ and let $(f^{\bullet M c_f}, x^{\bullet M d_x}) \in \mathcal{N}_2^{\mathbf{a}}$. For every m such that $0 < m \leq M(c_f - c_f^-)$, the edge $(f^{\bullet(M c_f - m)}, x^{\bullet(M d_x - m)})$ is an augmenting path of $\mathcal{N}_2^{\mathbf{a}}$ with respect to system $F^{\mathbf{a}}$.

Proof Consider Statement 1. If $c_f \leq c_f^-$ for every $f \in G$, then matching $\mathcal{N}_2$ is of maximal cardinality for F, so that no augmenting path exists. From now on, we assume $c_f > c_f^-$ for some $f \in G$. Select any such f and let $(f^{\bullet c_f}, x^{\bullet d_x})$ be the edge of $\mathcal{N}_2$ involving f. Then, $f^{\bullet(c_f - 1)}$ and $x^{\bullet(d_x - 1)}$ are involved in F, but $x^{\bullet(d_x - 1)}$ is not a vertex of $\mathcal{N}_2$. Thus, for every $f \in G$ such that $c_f > c_f^-$, then $(f^{\bullet(c_f - 1)}, x^{\bullet(d_x - 1)})$ is an augmenting path of $\mathcal{N}_2$ in F. The same reasoning applies for $(f^{\bullet(c_f - m)}, x^{\bullet(d_x - m)})$, provided that $0 < m \leq c_f - c_f^-$. This proves Statement 1. The same reasoning applies for proving Satement 2. □

Lemma 80 *1. The following formulas define matchings of maximal cardinality for F and $F^{\mathbf{a}}$, respectively:*

$$\mathcal{M} = \left\{ \left(f^{\bullet(c_f - m)}, x^{\bullet(d_x - m)}\right) \;\middle|\; \left(f^{\bullet c_f}, x^{\bullet d_x}\right) \in \mathcal{N}_2, 0 < m \leq c_f - c_f^- \right\}$$

$$\mathcal{M}^{\mathbf{a}} = \left\{ \left(f^{\bullet M(c_f - m)}, x^{\bullet M(d_x - m)}\right) \middle|\; \left(f^{\bullet c_f}, x^{\bullet d_x}\right) \in \mathcal{N}_2, 0 < m \leq c_f - c_f^- \right\}$$

2. Consider the surjective mapping $\psi : \mathcal{M}^{\mathbf{a}} \mapsto \mathcal{M}$, defined by

$$\psi\left(f^{\bullet M(c_f - m)}, x^{\bullet M(d_x - m)}\right) = \left(f^{\bullet(c_f - m)}, x^{\bullet(d_x - m)}\right)$$

and define

$$\psi_f : F^{\mathbf{a}} \mapsto F \quad \text{by} \quad \psi_f(f^{\bullet M(c_f - m)}) = f^{\bullet(c_f - m)} \ ,$$
$$\psi_x : X(F^{\mathbf{a}}) \mapsto X(F) \quad \text{by} \quad \psi_x(x^{\bullet M(d_x - m)}) = x^{\bullet(d_x - m)} \ .$$

Then, ψ preserves the Dulmage-Mendelsohn decompositions of $\mathcal{G}_F$ and $\mathcal{G}_{F^{\mathbf{a}}}$, meaning that $g \in \beta^{\mathbf{a}}_{\mathfrak{D}/\mathfrak{E}/\mathfrak{U}}$ if and only if $\psi_f(g) \in \beta_{\mathfrak{D}/\mathfrak{E}/\mathfrak{U}}$, and $y \in \beta^{\mathbf{a}}_{\mathfrak{D}/\mathfrak{E}/\mathfrak{U}}$ if and only if $\psi_x(y) \in \beta_{\mathfrak{D}/\mathfrak{E}/\mathfrak{U}}$.

Proof Statement 1 is a direct consequence of Lemma 79 and (174). We thus focus on Statement 2. By Statement 1, we can see that $(f^{\bullet(c_f-m)}, x^{\bullet(d_x-m)}) \in \mathcal{M}$ if and only if $(f^{\bullet M(c_f-m)}, x^{\bullet M(d_x-m)}) \in \mathcal{M}^{\mathbf{a}}$. Let $g = f^{\bullet(c_f-m)}$ be a vertex of $\mathcal{G}_F$ associated with a function. Then, g is unmatched in $\mathcal{M}$ if and only if $\psi_f(g)$ is unmatched in $\mathcal{M}^{\mathbf{a}}$. We conclude by invoking Lemma 78: the mappings ψ_x and ψ_f preserve the overdetermined blocks. A similar reasoning holds regarding the preservation of underdetermined blocks, by starting from unmatched vertices of variable type. □

This finishes the proof of Theorem 74.

Comparing the standardizations

The independence of the standardization with respect to the particular semantics of the derivative is immediate within long (continuous) modes. We thus focus on mode changes, where we must target a standard discrete-time dynamical system indexed by $\mathbb{N}$. Throughout this section, Assumptions 6 (Page 73), 8 (Page 83) and 9 (Page 86) are in force. We want to compare the respective standardizations of the returns of Algorithm 10 (ExecRun), when using the two expansions (164) and (165) for the derivatives.

We first adapt the generic system (124), computing the restart at a mode change:

$$\text{expand } X' \text{ as } \tfrac{1}{\partial}\textstyle\sum_M^N \alpha_n (X^\bullet - X)^{\bullet n} \text{ in } \ 0 = \mathbf{H}(X', X^\bullet, V, X) \ , \tag{175}$$
$$\text{also written } \ 0 = H(Z, X, \partial) \ . \tag{176}$$

In (175), X collects the state variables, while V collects the algebraic dependent variables, decomposed into impulsive and non-impulsive variables: $V = W^{\mathsf{i}} \cup W$. Since $N \geq 0$, the dependent variables of system (175) are $Z =_{\text{def}} (X^{\bullet(N+1)}, V)$, and we set $Y =_{\text{def}} (X^{\bullet(N+1)}, W)$; in words, Y collects the non-impulsive dependent variables.

In (175), the functions F and G are standard; so is the forward shift $x \mapsto x^\bullet$, since it will be mapped to a corresponding forward shift in the discrete time index $\mathbb{N}$. Hence, the infinitesimal time step ∂, arising in the expansion of the derivative in F, is the only reason for System (175) to be nonstandard.

Assumption 8 states that impulsive variables can be eliminated from system (175), resulting in the following reduced system:

$$\mathbf{K}(X', X^\bullet, W, X) = 0 \iff \exists W^{\mathsf{i}}.\mathbf{H}(X', X^\bullet, V, X) = 0 \ . \tag{177}$$

Recall that ∂ enters System (177)-left, i.e., the left-hand part of the equivalence given by Equation (177), via the expansion used for X' in (175). As part of Assumption 8:

$$\text{System (177)-left remains structurally nonsingular when } \partial = 0. \tag{178}$$

In (177), $(X^{\bullet(N+1)}, W)$ standardizes as the right-limit (X^+, W^+) whereas all the $X^{\bullet n}, n = P, \ldots, 0$ standardize as the left-limit X^-. Therefore, in the expansion used for X' in (175), all terms cancel out but the last one. Hence, System (177) standardizes as:

- Case $N > 0$: substitute $\partial \leftarrow 0$ in $\mathbf{K}\left(\frac{\alpha_N}{\partial}(X^+ - X^-), X^-, W, X\right) = 0$;
- Case $N = 0$: substitute $\partial \leftarrow 0$ in $\mathbf{K}\left(\frac{\alpha_N}{\partial}(X^+ - X^-), X^+, W, X\right) = 0$.

The difference lies in the X^- vs. X^+, highlighted in red. These substitutions are both legitimate, thanks to (178). In both cases, the result does not depend on α_N, since $\alpha_N \neq 0$. Hence, we get the following result:

Theorem 81 *Under Assumptions 6, 8, and 9, the standardization of the restart conditions does not depend on the expansion (165) for the derivative.*

Comment 82 So far, Theorem 81 applies to mDAE systems possessing only long modes. We could extend this theorem in several directions. First, we could include in our modeling language left- or right-limits: x^- or x^+. Then, our reasoning would apply with no change provided that x^- and x^+ are encoded as ${}^\bullet x$ or $x^\bullet$ if expansion (164) is used, and ${}^{\bullet(M+1)}x$ or $x^{\bullet(M+1)}$ if expansion (165) is used. Removing Assumption 9 could be handled similarly, by redefining transient modes as modes of nonstandard duration $M\partial$ if expansion (165) is used, instead of ∂ if expansion (164) is used.

11.2 A correctness result

Our approach, consisting of the structural analysis in the nonstandard domain (Algorithm 10 and its refined version Algorithm 13) followed by standardization, either fails, in which case it returns proper diagnosis, or succeeds and produces executable simulation code. The natural question is:

Question 83 Does this approach compute the solution of the given mDAE system?

Once again, answering Question 83 in its full generality is out of reach as, to the best of our knowledge, there is no general mathematical definition of the solution of an mDAE system. Pathologies may exist in the complement of the long 'continuous' modes: possibly unbounded cascades of events, chattering or sliding modes, Zeno behaviors for events of mode changes or transient modes, and more. Even when the set of events of mode changes is 'gentle' (a finite or diverging sequence of instants), the characterization of the impulses may not be available. Still, as mentioned in the Introduction, notions of solutions of mDAE systems exist for some restricted classes.

In this section, we provide an answer to Question 83 for a restricted subclass of mDAE systems, by reproducing the results of [8] and giving their detailed proofs. For the description of the class of systems considered, we use the notion of function of bounded variation: a function $f : \mathbb{R} \to \mathbb{R}$ has *bounded variation* if it is the primitive of a Lebesgue integrable function [26]. As a consequence, if f has bounded variation, then

$$\lim_{h \searrow 0} \int_t^{t+h} f'(s)ds \quad = \quad 0 \ . \tag{179}$$

Definition 84 (semilinear systems) *Call* semilinear *an mDAE system S such that, for each mode μ, the active DAE system takes the restricted form*

$$\begin{cases} 0 = A(X_s)X' + B_\mu(X) \\ 0 = C_\mu(X) \end{cases} \tag{180}$$

where:

1. *X_s collects the smooth elements of X, i.e., those being continuous and of bounded variation around each instant of mode change. Other elements of X might be discontinuous.*
2. *Both matrix $A(.)$ and the mode-dependent vectors $B_\mu(.)$ and $C_\mu(.)$ are continuous functions of their arguments, and remain bounded in a neighborhood of each instant of mode change.*

3. *The matrix*

$$\mathbf{J} = \begin{bmatrix} A(X_s) \\ \mathbf{J}_X C_\mu(X) \end{bmatrix}$$

is regular, where $\mathbf{J}_X C_\mu(X)$ *denotes the Jacobian matrix of* C_μ *with respect to* X*, evaluated at* X*.*

Comment 85 The special form (180) and Condition 3 for the Jacobian matrix express that mode dynamics have been processed using the Σ-method: differentiating the algebraic constraint once yields a regular linear system determining X'. Then, Conditions 1 and 2 together make the impulse analysis easier.

Comment 86 In [8], Section 4.3, it is shown that multi-body systems with contacts yield semi-linear mDAE systems.

For semilinear mDAE systems, the solution at an instant t_* of mode change can be determined as follows. After proper reordering of the entries of X, we decompose vector X into its smooth and nonsmooth parts and decompose matrix A accordingly:

$$X = \begin{bmatrix} X_s \\ X_n \end{bmatrix} \quad \text{and} \quad A = \begin{bmatrix} A_s & A_n \end{bmatrix} .$$

Since X_s is smooth at any instant, by Condition 1, matrix $A\big(X_s(t_*)\big)$ is known prior to computing the mode change. In contrast, $X_n'(u)du$ is a Dirac measure at mode change.

To evaluate the restart values for all elements of X, we integrate the system dynamics over the interval $[t_* - \varepsilon, t_* + \varepsilon]$ for $\varepsilon > 0$ sufficiently small, taking into account the dynamics before and after the mode change. In the following calculations, μ^- and μ^+ denote the mode prior and after the mode change, and B_+ and C_+ stand for B_{μ^+} and C_{μ^+}, respectively.

$$\begin{aligned}
& \int_{[t_*-\varepsilon,t_*)} \big(A(X_s(u))X'(u)+B_-(X(u))\big)du + \int_{[t_*,t_*+\varepsilon]} \big(A(X_s(u))X'(u)+B_+(X(u))\big)du \\
= \quad & \int_{[t_*-\varepsilon,t_*)} \big(A_s(X_s(u))X_s'(u)+B_-(X(u))\big)du + \int_{[t_*,t_*+\varepsilon]} \big(A_s(X_s(u))X_s'(u)+B_+(X(u))\big)du \\
& + \int_{[t_*-\varepsilon,t_*+\varepsilon]} A_n(X_s(u))X_n'(u)du \\
\approx \quad & 0 \;+\; 0 \;+\; A_n(X_s(t_*))\big(X_n(t_*+\varepsilon) - X_n(t_*-\varepsilon)\big) \\
\approx \quad & A(X_s(t_*))\big(X(t_*+\varepsilon) - X(t_*-\varepsilon)\big)
\end{aligned}$$

The evaluation to zero of the first two integrals follows from (179). For the third integral, we use the fact that $X_n'(u)du$ is approximately a Dirac measure at t_* in the indicated interval. This reasoning leads to the following result determining the restart conditions at t_*:

Theorem 87 ([8]) *The restart conditions* X^+ *are determined from the left-limits* X^- *by using the following system of equations:*

$$\begin{cases} 0 = A(X_s^-)(X^+ - X^-) \\ 0 = C_+(X^+) \end{cases} \tag{181}$$

which is locally regular thanks to Condition 3 of Definition 84.

Whether we can solve system (181) depends on the particular system at hand. It remains to compare scheme (181) with our approach. The nonstandard semantics of (180) is:

$$\begin{cases} 0 = A(X_s)(X^\bullet - X) + \partial \times B_\mu(X) \\ 0 = C_\mu(X) \end{cases}$$

and applying the Σ-method to it yields the augmented system

$$\left\{\begin{array}{l} 0 = A(X_s)(X^\bullet - X) + \partial \times B_\mu(X) \\ 0 = C_\mu(X) \\ 0 = C_\mu(X^\bullet) \end{array}\right. . \tag{182}$$

The reasoning performed at Section 7.2 applies to semilinear mDAE systems. Following (77), we unfold (182) at the two successive instants ${}^\bullet t_*$ and t_*:

$$\begin{array}{l} {}^\bullet t_* : \left\{\begin{array}{l} 0 = A({}^\bullet X_s)(X - {}^\bullet X) + \partial \times B_-({}^\bullet X) \\ 0 = C_-({}^\bullet X) \\ 0 = C_-(X) \end{array}\right. \\ t_* : \left\{\begin{array}{l} 0 = A(X_s)(X^\bullet - X) + \partial \times B_+(X) \\ 0 = C_+(X) \\ 0 = C_+(X^\bullet) \end{array}\right. \end{array} \tag{183}$$

Eliminating ${}^\bullet X$ in (183) yields

$$\left\{\begin{array}{ll} 0 = C_-(X) & (e^-) \\ 0 = A(X_s)(X^\bullet - X) + \partial \times B_+(X) & (e_1^+) \\ 0 = C_+(X) & (e_2^+) \\ 0 = C_+(X^\bullet) & (e_3^+) \end{array}\right.$$

Equations (e^-) and (e_2^+) are in conflict: we discard the latter. This finally yields

$$\left\{\begin{array}{ll} 0 = C_-(X) & (e^-) \\ 0 = A(X_s)(X^\bullet - X) + \partial \times B_+(X) & (e_1^+) \\ 0 = C_+(X^\bullet) & (e_3^+) \end{array}\right. \tag{184}$$

In System (184), (e^-) is a context, whereas (e_1^+) and (e_3^+) determine $X^\bullet$ as a function of X. Setting $\partial \leftarrow 0$ in (e_1^+, e_3^+) yields a structurally nonsingular system by Condition 3 of Definition 84. Hence, by Theorem 67, setting $\partial \leftarrow 0$ in (e_1^+, e_3^+) performs the standardization: the scheme of Theorem 87 is recovered.

Theorem 88 (correctness result) *For semilinear mDAE systems, our approach computes the correct solution.*

Comment 89 Our approach computes solutions for any mDAE system whose structural analysis (Algorithm 10 or Algorithm 13) succeeds. This is way beyond the class of semilinear mDAE systems. It would be desirable to compare the so obtained scheme with schemes known for more dedicated physics. □

Comment 90 Hilding Elmqvist and Martin Otter have developed the ModiaMath tool for semilinear mDAE systems by effectively implementing scheme (181). □

11.3 Numerical scheme for restarts

In this section, we focus on the system of equations defining restarts, in its form (125), which we recall for convenience:

$$H(Z, X, \partial) = 0 \ . \tag{185}$$

Our aim is to justify our claim that, if we remain 'blind' with respect to impulsive variables, the solution of this system is well approximated by the standard system

$$H(Z, X, \delta) = 0 \ , \tag{186}$$

where the infinitesimal ∂ has been replaced by a small standard parameter $\delta > 0$. This is essentially the proposed numerical scheme. Throughout this section, Assumptions 6 and 8 of Section 10.5 are in force.

We briefly recall the corresponding notations for convenience: in (185), X collects the state variables and $Z =_{\text{def}} (X^\bullet, V)$ collects the dependent variables. The algebraic variables V can be split into impulsive and non-impulsive variables: $V = W^{\text{i}} \cup W$, and we set $Y =_{\text{def}} (X^\bullet, W)$. Assumption 8 states that impulsive variables can be eliminated from System (186), resulting in the reduced system

$$K(Y, X, \delta){=}0 \ , \tag{187}$$

that remains structurally nonsingular when $\delta = 0$.

Theorem 91 *The following statements hold:*

1. *Let Y_* be a solution of (187). There exists a selection map $\delta \mapsto (Y(\delta), W^{\text{i}}(\delta))$, from $\mathbb{R}_{>0}$ to the set of solutions of the system (186), such that $\lim_{\delta \searrow 0} Y(\delta){=}Y_*$.*
2. *Conversely, let $(Y(\delta), W^{\text{i}}(\delta))$ be a solution of (186) such that $Y_* =_{\text{def}} \lim_{\delta \searrow 0} Y(\delta)$ exists; then, Y_* is a solution of (187).*

Proof We successively prove the two statements.

Statement 1. By the Implicit Function Theorem applied to the standard function K, there exists a function $(\hat{X}, \delta) \mapsto Y(\hat{X}, \delta)$ of class $\mathcal{C}^1$ such that $Y(X, 0){=}Y_*$ and $K(Y(\hat{X}, \delta), \hat{X}, \delta){=}0$ holds for $(\hat{X}, \delta)$ ranging over a neighborhood of $(X, 0)$. Then, by Assumption 8, $\exists W^{\text{i}}.H(Z, X, \delta){=}0 \iff K(Y, X, \delta){=}0$ holds, which concludes the proof of this statement.

Statement 2. This is trivial and does not use Assumption 8. Since K is smooth, the two conditions (1) $K(Y(\delta), X, \delta){=}0$ for every $\delta > 0$, and (2) $Y(\delta)$ converges to Y_*, imply $K(Y_*, X, 0){=}0$. □

Corollary 92 *Let Assumption 8 be in force and let the systems (186) and (187) possess a unique solution for Z and Y, respectively. Then, $H(Z(\delta), X, \delta){=}0$ implies $K(\lim_{\delta \searrow 0} Y(\delta), X, 0){=}0$.*

For its practical application, this scheme requires the prior knowledge of the decomposition $V = W^{\text{i}} \cup W$ of algebraic variables into impulsive and non-impulsive ones: we solve System (186) for Z and simply discard the values of W^{i} from this solution. The drawback of this scheme is that system (186) will be likely ill-conditioned.

Thus, in practice, we would recommend a rescaling of the impulsive variables based on the impulsion order: for every impulsive variable w or order $\mu =_{\text{def}} [\![w]\!]$, we perform the substitution $w \leftarrow \nu \times \delta^{-\mu}$, where ν is the new variable replacing w—such a rescaling works only for variables of finite impulsion order, see Section 10.1.

11.4 Experimental results: mode changes of a nonsemilinear system

To exercise a numerical method computing state-jumps at mode changes, when some variables are impulsive, let us consider the following multimode DAE system:

$$\left\{\begin{array}{rll} & & \omega_1' = a_1\omega_1 + b_1\tau_1^3 \quad (e_1) \\ & & \omega_2' = a_2\omega_2 + b_2\tau_2 \quad (e_2) \\ \texttt{if } \gamma & \texttt{then} & \omega_1 - \omega_2 = 0 \quad (e_3) \\ & \texttt{and} & \tau_1 + \tau_2 = 0 \quad (e_4) \\ \texttt{if not } \gamma & \texttt{then} & \tau_1 = 0 \quad (e_5) \\ & \texttt{and} & \tau_2 = 0 \quad (e_6) \end{array}\right. \tag{188}$$

This system is identical to the clutch model (3), with the only difference that torque τ_1 appears non-linearly in equation (e_1). Therefore, this system is not semilinear, and the direct approach based on linear-algebraic techniques used in Section 11.2 does not apply. Nevertheless, we shall demonstrate on this example that our general approach applies. We focus on the mode change $\gamma : \text{F} \to \text{T}$. First, we perform impulse analysis following Section 10.1. Based on this impulse analysis, we apply a change of variable that bypasses the need for eliminating impulsive variables. We then standardize the resulting algebraic system of equations and we prove that it possesses a unique solution.

Using the structural analysis method detailed in Section 2.4, the state-jump is solution of the following system of algebraic equations, where $\omega_1^\bullet$, $\omega_2^\bullet$, τ_1 and τ_2 are the unknowns:

$$\begin{cases} 0 = \frac{\omega_1^\bullet - \omega_1}{\partial} - a_1\omega_1 - b_1\tau_1^3 & (e_1^\partial) \\ 0 = \frac{\omega_2^\bullet - \omega_2}{\partial} - a_2\omega_2 - b_2\tau_2 & (e_2^\partial) \\ 0 = \omega_1^\bullet - \omega_2^\bullet & (e_3^\bullet) \\ 0 = \tau_1 + \tau_2 & (e_4) \end{cases} \tag{189}$$

Applying the method of Section 10.1, the impulse orders of the dependent variables $\omega_1^\bullet - \omega_1, \omega_2^\bullet - \omega_2, \tau_1, \tau_2$ are solution of the following system of equalities and inequalities:

$$\begin{cases} 1 + [\![\omega_1^\bullet - \omega_1]\!] \leq \max\{[\![a_1\omega_1]\!], [\![b_1]\!] + 3[\![\tau_1]\!]\} \\ [\![a_1\omega_1]\!] \leq \max\{1 + [\![\omega_1^\bullet - \omega_1]\!], [\![b_1]\!] + 3[\![\tau_1]\!]\} \\ [\![b_1]\!] + 3[\![\tau_1]\!] \leq \max\{1 + [\![\omega_1^\bullet - \omega_1]\!], [\![a_1\omega_1]\!]\} \\ 1 + [\![\omega_2^\bullet - \omega_2]\!] \leq \max\{[\![a_2\omega_2]\!], [\![b_2]\!] + [\![\tau_2]\!]\} \\ [\![a_2\omega_2]\!] \leq \max\{1 + [\![\omega_2^\bullet - \omega_2]\!], [\![b_2]\!] + [\![\tau_2]\!]\} \\ [\![b_2]\!] + [\![\tau_2]\!] \leq \max\{1 + [\![\omega_2^\bullet - \omega_2]\!], [\![a_2\omega_2]\!]\} \\ [\![\omega_1^\bullet - \omega_1]\!] = [\![\omega_2^\bullet - \omega_2]\!] = 0 \\ [\![\tau_1]\!] = [\![\tau_2]\!] \end{cases} \tag{190}$$

System (190) admits a unique solution: $[\![\omega_1^\bullet - \omega_1]\!] = [\![\omega_2^\bullet - \omega_2]\!] = 0$ and $[\![\tau_1]\!] = [\![\tau_2]\!] = \frac{1}{3}$. This impulse analysis tells us that if a solution to System (189) exists, then unknowns τ_1 and τ_2 are of order $O(\partial^{-\frac{1}{3}})$. The following substitution helps solving System (189): $\tau_i = \partial^{-\frac{1}{3}}\hat{\tau}_i$. With this substitution, System (189) becomes:

$$\begin{cases} 0 = \omega_1^\bullet - \omega_1 - \left(\partial a_1\omega_1 + b_1\hat{\tau}_1^3\right) & (e_1^\partial) \\ 0 = \omega_2^\bullet - \omega_2 - \left(\partial a_2\omega_2 + \partial^{\frac{2}{3}} b_2\hat{\tau}_2\right) & (e_2^\partial) \\ 0 = \omega_1^\bullet - \omega_2^\bullet & (e_3^\bullet) \\ 0 = \hat{\tau}_1 + \hat{\tau}_2 & (e_4) \end{cases} \tag{191}$$

Let us replace each unknown by its unique decomposition into a standard part and an infinitesimal part. Regarding velocities, we have: $\omega_i^\bullet = \bar{\omega}_i^\bullet + \phi_i$, where $\phi_i \approx 0$, for $i = 1, 2$. Next, since the rescaled variables $\hat{\tau}_i$ are non-impulsive, the same substitution is legitimate by Lemma 59: $\hat{\tau}_i = \bar{\tau}_i + \psi_i$, with $\psi_i \approx 0$, for $i = 1, 2$. System (191) then becomes:

$$\begin{cases} \bar{\omega}_1^\bullet + \phi_1 - \omega_1 = \partial a_1\omega_1 + b_1\left(\bar{\tau}_1^3 + 3\bar{\tau}_1^2\psi_1 + 3\bar{\tau}_1\psi_1^2 + \psi_1^3\right) & (e_1^\partial) \\ \bar{\omega}_2^\bullet + \phi_2 - \omega_2 = \partial a_2\omega_2 + \partial^{\frac{2}{3}} b_2\left(\bar{\tau}_2 + \psi_2\right) & (e_2^\partial) \\ \bar{\omega}_1^\bullet + \phi_1 - \bar{\omega}_2^\bullet - \phi_2 = 0 & (e_3^\bullet) \\ \bar{\tau}_1 + \psi_1 + \bar{\tau}_2 + \psi_2 = 0 & (e_4) \end{cases} \tag{192}$$

Zeroing the infinitesimals in these equations yields the following nonsingular system of equations, which, by Theorem 67, is then the standardization of System (192):

$$\begin{cases} \bar{\omega}_1^\bullet - \omega_1 = b_1\bar{\tau}_1^3 & (\bar{e}_1^\partial) \\ \bar{\omega}_2^\bullet - \omega_2 = 0 & (\bar{e}_2^\partial) \\ \bar{\omega}_1^\bullet - \bar{\omega}_2^\bullet = 0 & (\bar{e}_3^\bullet) \\ \bar{\tau}_1 + \bar{\tau}_2 = 0 & (\bar{e}_4) \end{cases} \tag{193}$$

This standard system of equations admits a unique solution:

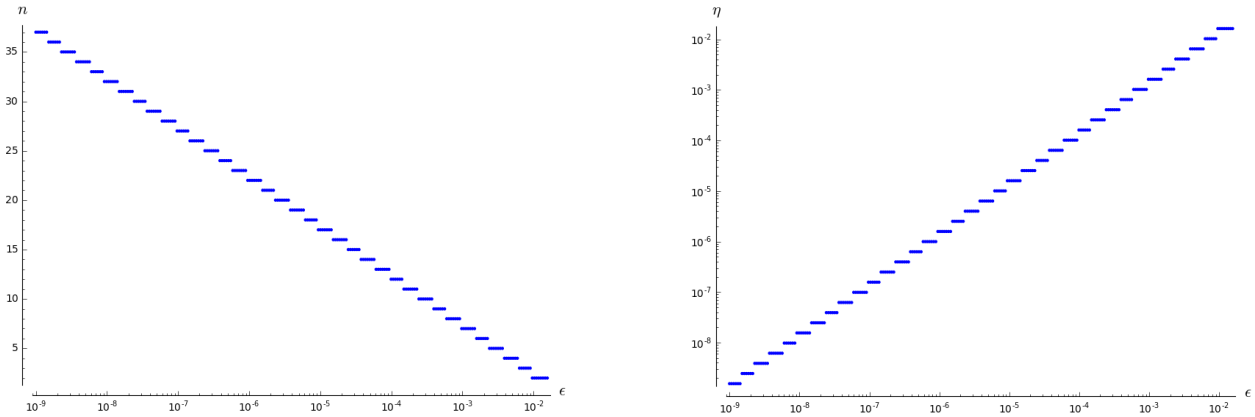
$$\bar{\omega}_1^\bullet = \bar{\omega}_2^\bullet = \omega_2 \quad \text{and} \quad \bar{\tau}_1 = -\bar{\tau}_2 = \sqrt[3]{\frac{\omega_2 - \omega_1}{b_1}}\,. \tag{194}$$

We now show that solving System (189) with a small standard positive parameter instead of the nonstandard ∂ yields a numerical scheme converging to the solution (194), if we disregard impulsive variables. Algorithm 14 describes our numerical method, whereas Figure 27 displays the results.

Algorithm 14 Numerical method for approximating the state-jump of non-semilinear system (188) for mode change $\gamma : \text{F} \to \text{T}$; $F_h(u)$ denotes the value of the right-hand side of System (189) for a 4-tuple u of values for the unknowns and ∂ replaced by h.

Require: Difference equations $F_h(u) : \mathbb{R}\times\mathbb{R}^N\to\mathbb{R}^N$, $u_0\in\mathbb{R}^N$ initial guess, $\pi : \mathbb{R}^N\to\mathbb{R}^M$ projection on the set of non-impulsive variables, $h_0>0$ initial time step, $\epsilon>0$ tolerance bound, $0<\theta<1$ scaling factor ; **return** u estimated state-jump

1: $u \leftarrow u_0$; $x \leftarrow \pi(u_0)$; $h \leftarrow h_0$
2: **repeat**
3: $\quad y \leftarrow x$
4: $\quad u \leftarrow \text{NSolve}(\text{system} = F_h, \text{guess} = u, \text{tolerance} = h)$
5: $\quad x \leftarrow \pi(u)$; $h \leftarrow \theta h$
6: **until** $|x - y|_\infty \leq \epsilon$



(a) (b)

Figure 27: Numerical method for computing the state-jump of non-semilinear system (188): (a) number of iterations as a function of ϵ; (b) error η as a function of ϵ.

Experimental results on this method have been obtained using Sage. The nonlinear solver used is the least squares solver of the Scipy's `optimize` library. Figure 27 gives, on the left, the number of iterations required, and on the right, the absolute error $\eta = |x - \bar{x}|_\infty$ between the computed solution and the exact solution $\bar{x} = (\omega_2, \omega_2)$. The scaling factor is $\theta = 0.5$, the initial time step $h_0 = 10^{-2}$ and $u_0 = (\omega_1, \omega_2, 0, 0)$. It can be observed that the method converges in not more than 37 iterations, down to $\epsilon = 10^{-9}$. It should also be remarked that, experimentally, the absolute error is bounded : $\eta < 2\epsilon$.

12 Toward a tool supporting our approach

The general approach we presented sets formidable challenges for a tool development. We identify three important requirements for such a tool.

- First, it should implement the theory: structural analysis must be adjusted to every different mode and mode change. Both aspects are included in what we call a 'multimode structural analysis'.
- Second, the described approach aims at the compilation of mDAE systems. As such, 'on-the-fly' techniques for the structural analysis of such systems, as advocated in [17] and [27], cannot be taken into consideration. As soon as a model is structurally unsound in any given mode or at any given mode change, it should be rejected at compile time. For accepted models, the output of the tool should enable the generation of efficient simulation code.
- Third, enumerating all modes and mode changes to perform multimode structural analysis is bound to collapse. An mDAE compiler should be able to handle models in a clever way to avoid such enumeration.

We believe that we achieved an important first step towards alleviating these challenges with our tool IsamDAE under development. IsamDAE implements the structural analysis of all modes of an mDAE without any enumeration of the modes, instead relying on Binary Decision Diagrams (BDD, see for instance [18]) for implicit representations of the mode-dependent structure of an mDAE. This is described more thoroughly in [19, 20].

In order to illustrate how a multimode model is currently handled by IsamDAE, this Section focuses on the so-called RLDC2 model, a simple electronic circuit shown in Figure 28. This example, provided to us by Sven-Erik Mattsson, is actually a difficult one to handle for the existing Modelica tools because of its mode-dependent index and structure, even though it exhibits no impulsive behavior. Indeed, we show once more how this model is mishandled by two leading Modelica tools, namely, Dymola and OpenModelica, and hint at reasons for this fact, thereby justifying our work once again. We then detail its handling by the IsamDAE tool, in order to provide an insight about why this tool could be expanded into a complete multimode structural analysis suite.

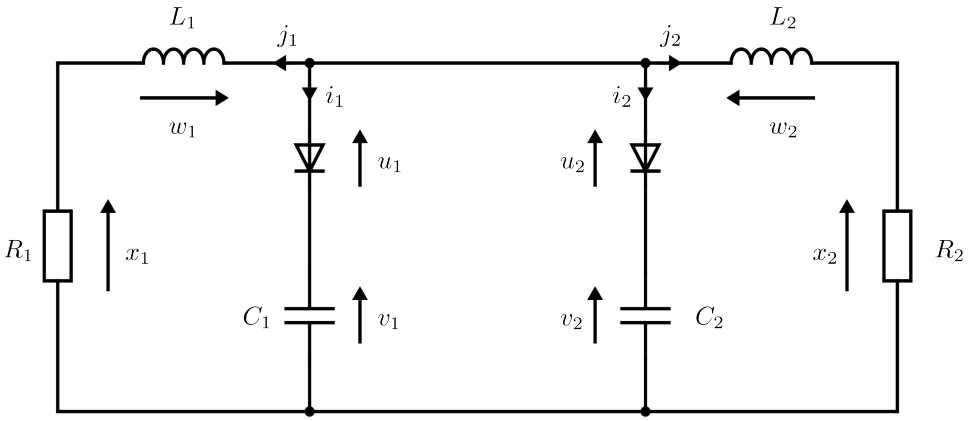


Figure 28: Schematics of the RLDC2 circuit.

12.1 The RLDC2 model

The circuit consists in two RLC circuits interconnected in parallel and in which two diodes have been inserted. The diodes are considered to be ideal, meaning that they are not ruled by the customary Shockley law $i = I\left(e^{u/U} - 1\right)$, but rather by the complementarity condition $0 \leq i \perp -u \geq 0$, meaning that, at any moment, both i and $-u$ are nonnegative and $iu = 0$ (Section 1.2.2 already mentions complementarity conditions). Figure 29 illustrates these two diode laws.

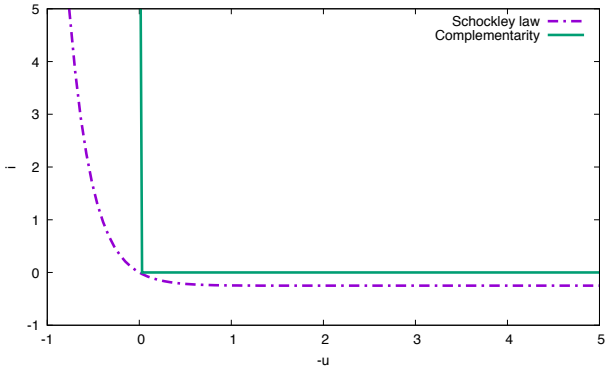


Figure 29: Shockley law (dashed, magenta) vs. ideal complementarity condition (solid, green).

As in Section 1.2.2, we redefine the graph of this complementarity condition as a parametric curve, represented by the following three equations:

$$\begin{array}{rcll} s & = & \text{if } \gamma \text{ then } i \text{ else } -u & (S) \\ 0 & = & \text{if } \gamma \text{ then } u \text{ else } i & (Z) \\ \gamma & = & (s \geq 0) & (G) \end{array}$$

To avoid a logico-numerical fixpoint, we consider instead the system obtained by replacing s with its left-limit s^- in equation (G):

$$\begin{array}{rcll} s & = & \text{if } \gamma \text{ then } i \text{ else } -u & (S) \\ 0 & = & \text{if } \gamma \text{ then } u \text{ else } i & (Z) \\ \gamma & = & (s^- \geq 0) & (G^-) \end{array}$$

Under the assumption that u and i are continuous functions of time (which turns out to be a valid assumption for the RLDC2 circuit), $s(t) = s^-(t)$ holds at every instant t. This implies that systems $(S), (Z), (G)$ and $(S), (Z), (G^-)$ are equivalent. Using this encoding of the complementarity condition, the RLDC2 circuit is modeled as the mDAE shown in Figure 30, with two Boolean variables γ_1 and γ_2 and four multimode equations (S_1), (Z_1), (S_2) and (Z_2), incident to a set of variables that depends on the mode of the system (i.e., the values of Boolean variables γ_1 and γ_2). Note that this model belongs to the class introduced in Definition 33.

12.2 Handling the RLDC2 model with Modelica tools

The Modelica model of the RLDC2 circuit is given in Figure 31; it is a direct translation of the model above. The authors had the opportunity to test it on two implementations of the Modelica language: OpenModelica v1.12[26] and Dymola 2019.[27]

Identical results are obtained with both tools: the model is deemed nonsingular at compile time, yet a runtime error immediately occurs (at time $t = 0$) during simulation, as shown in Figure 32.

The issue is that the structural analysis implemented in these tools treats the model as a single-mode DAE, disregarding the mode-dependent variability of the incidence relations. The consequence is that, despite the model being deemed structurally nonsingular by these tools, blocks of equations that are structurally singular in some modes are produced. This immediately leads to runtime errors: both tools attempt a pivoting of the Jacobian matrix by an element that is equal to zero in some modes.

$$
\begin{array}{rcll}
0 & = & i_1 + i_2 + j_1 + j_2 & (K_1) \\
x_1 + w_1 & = & u_1 + v_1 & (K_2) \\
u_1 + v_1 & = & u_2 + v_2 & (K_3) \\
u_2 + v_2 & = & x_2 + w_2 & (K_4) \\
w_1 & = & L_1 \cdot j_1' & (L_1) \\
w_2 & = & L_2 \cdot j_2' & (L_2) \\
i_1 & = & C_1 \cdot v_1' & (C_1) \\
i_2 & = & C_2 \cdot v_2' & (C_2) \\
x_1 & = & R_1 \cdot j_1 & (R_1) \\
x_2 & = & R_2 \cdot j_2 & (R_2) \\
s_1 & = & \text{if } \gamma_1 \text{ then } i_1 \text{ else } -u_1 & (S_1) \\
s_2 & = & \text{if } \gamma_2 \text{ then } i_2 \text{ else } -u_2 & (S_2) \\
0 & = & \text{if } \gamma_1 \text{ then } u_1 \text{ else } i_1 & (Z_1) \\
0 & = & \text{if } \gamma_2 \text{ then } u_2 \text{ else } i_2 & (Z_2) \\
\gamma_1 & = & (s_1^- \geq 0) & (\gamma_1^-) \\
\gamma_2 & = & (s_2^- \geq 0) & (\gamma_2^-)
\end{array}
\tag{195}
$$

Figure 30: The RLDC2 model (*n.b.*: variable y' denotes the time derivative of y).

12.3 Structural analysis of the RLDC2 model

The mDAE system of Figure 30 has two modes of nonzero index, when diodes are either both passing or both open. Having performed index reduction on each of them, we show the corresponding block dependency graphs in Figures 33 and 34, respectively—see Lemma 14 and Figure 19. These two graphs significantly differ, highlighting the fact that a mode-dependent structural analysis is necessary.

Figure 35 shows a graph generated by the structural analysis performed by our software IsamDAE [19, 20], for the mDAE system of Figure 30. This graph does not consist of one graph per mode, but, rather, of a set of edges labeled by a propositional formula characterizing the set of modes in which the considered branch is involved. This data structure prevents the combinatorial explosion arising in mDAE systems composed of a large number of mDAE subsystems, when modes are enumerated.

The block triangular form of the system can then be found at runtime for any given mode, for instance by evaluating the block dependency graph in this mode and performing a topological sort on the resulting graph. For the RLDC2 example, when evaluated in the mode in which both diodes are passing (resp. blocking), the graph of Figure 35 yields the dependency graph of Figure 33 (resp. Figure 34); the same holds in the remaining modes. All equation blocks can be turned into efficient simulation code at compile time, so that the computational overhead due to a mode switching at runtime is minimized.

12.4 Perspectives

As shown on the RLDC2 example above, our tool IsamDAE addresses the issue of performing the structural analysis of all modes of an mDAE while avoiding any explicit enumeration of these modes. Although the example detailed above is, for clarity purposes, a small one, IsamDAE is provably able to handle models with a very large number of modes, as detailed in [19, 20].

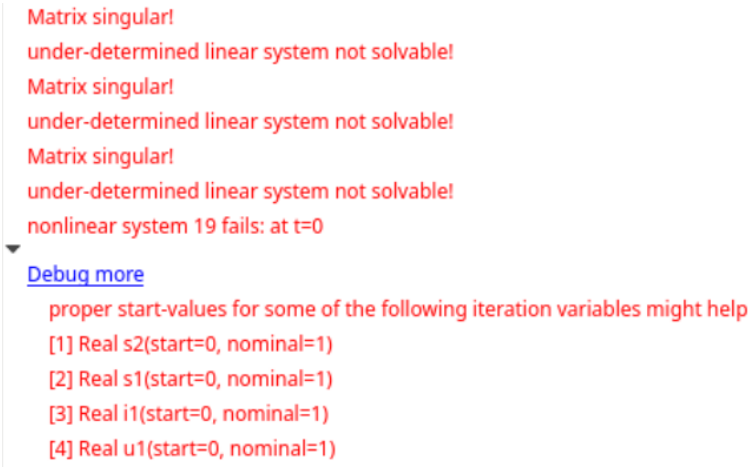
As such, the tool already provides the data structures and key algorithms required for addressing the second and third challenges invoked at the beginning of this section. This encourages us to implement, in the same tool, the structural analysis of mode changes, in the same 'all-at-once' fashion as for the modes themselves. This is indeed the main perspective of our work, as we believe

[26] https://openmodelica.org
[27] https://www.3ds.com/products-services/catia/products/dymola/

```
model RLDC2_CC "RLDC2 example with complementarity conditions"
  parameter Real R1 = 10.0;
  parameter Real R2 = 15.0;
  parameter Real L1 = 1.0;
  parameter Real L2 = 1.5;
  parameter Real C1 = 0.10;
  parameter Real C2 = 0.15;
  Real i1;
  Real i2;
  Real j1(start=2.0,fixed=true);
  Real j2(start=1.0,fixed=true);
  Real u1;
  Real u2;
  Real v1(start=0.5,fixed=true);
  Real v2(start=1.0,fixed=true);
  Real w1;
  Real w2;
  Real x1;
  Real x2;
  Real s1;
  Real s2;
  Boolean g1(start=false);
  Boolean g2(start=false);
equation
  0 = j1+i1+i2+j2; // (K1)
  x1+w1 = u1+v1; // (K2)
  u1+v1 = u2+v2; // (K3)
  u2+v2 = x2+w2; // (K4)
  x1 = R1*j1; // (R1)
  x2 = R2*j2; // (R2)
  w1 = L1*der(j1); // (L1)
  w2 = L2*der(j2); // (L2)
  i1 = C1*der(v1); // (C1)
  i2 = C2*der(v2); // (C2)
  s1 = if g1 then i1 else -u1; // (S1)
  s2 = if g2 then i2 else -u2; // (S2)
  0 = if g1 then u1 else i1; // (Z1)
  0 = if g2 then u2 else i2; // (Z2)
  g1 = (s1 >= 0); // (G1)
  g2 = (s2 >= 0); // (G2)
    annotation (...);
end RLDC2_CC;
```

Figure 31: Modelica model of the RLDC2 circuit.

(a)



(b)

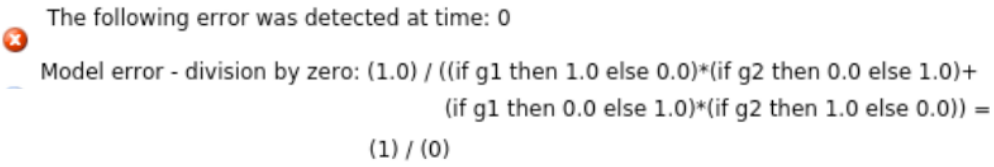


Figure 32: Error messages produced during simulation of the RLDC2 model with (a) OpenModelica and (b) Dymola, both at the initial time of the simulation.

it will be a crucial step towards the design of a mathematically sound compiler for multimode DAE systems.

13 Conclusion

We have proposed sound foundations for the compilation and code generation for physical systems modeling languages relying on DAE for their continuous dynamics; major instances are Modelica and Simscape. As its main contribution, our theory explicitly considers events of mode change and properly handles any multimode system having at most finite cascades of mode changes—sliding modes are not supported by our method.

A key step was the extension of structural analysis from single-mode to multimode systems. In this extension, the handling of mode changes was the main difficulty. We illustrated our approach on small examples that are not properly supported by existing tools (with the exception of Modia-Math). These examples exhibit mode-dependent index, which is not well supported by existing techniques.

For the clutch example, the most natural model consists in specifying the dynamics for the two modes "released" and "engaged"; then, the compiler automatically generates the restart conditions at mode changes. The "Westinghouse air brake" example motivated the consideration of assertions—not studied here but included in our tool under development. The Cup-and-Ball example is interesting in that, when the rope gets straight, inelastic or elastic impact can be hypothesized. This example pointed the issue of long vs. transient modes (modes having zero duration). Depending on whether the straight rope mode was assumed long (inelastic impact) or transient (elastic impact), the original model was accepted or rejected. For the latter, underspecification was detected and this information was returned to the user, who could refine his/her model by specifying an impact law. The so enriched model was accepted and code properly generated at mode change.

Generally, our method allows us to reject models on the basis of under/over-specification. Logico-numerical fixpoint equations (cases in which numerical variables determine the value of a Boolean condition that, in turn, guards the equation allowing to evaluate the same numerical

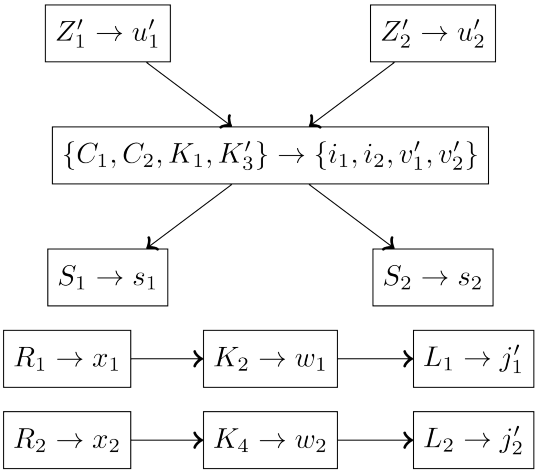


Figure 33: Block dependency graph for the RLDC2 example with both diodes passing (i.e., $\gamma_1 = \gamma_2 = \mathbf{T}$).

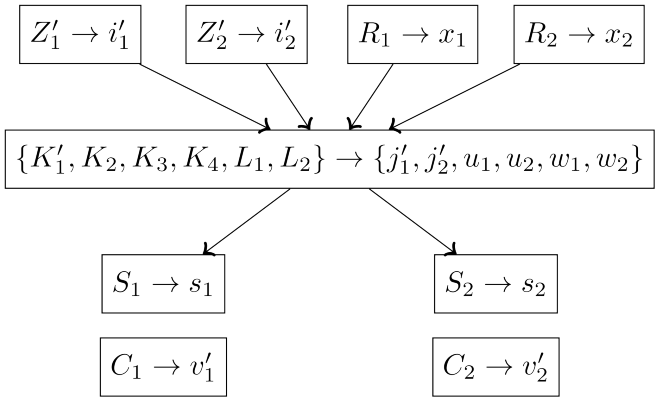


Figure 34: Block dependency graph for the RLDC2 example with both diodes blocking (i.e., $\gamma_1 = \gamma_2 = \mathbf{F}$).

variables) are prohibited by setting restrictions on the language syntax.

The structure of the code we generate is reminiscent of the so-called dummy derivatives method [37]: for both the continuous dynamics and the handling of transient modes or mode change events, we call an algebraic equation system solver for computing derivatives or next state values, as a function of the current state, while meeting algebraic constraints.

A first requirement on our theory was that it should support the analysis of any model related to any kind of physics—we aim at being physics agnostic. This support should hold regardless of the particular properties of the model, such as conditions ensuring existence and uniqueness of solutions. The motivation for this was that the user is not expected to check such conditions before submitting her/his model.

A second requirement was the ability to handle the conflicts that may result, at mode changes, between the dynamics of the previous mode and the consistency conditions implied by the new mode.

We do not see how these two requirements could be addressed without relying on nonstandard analysis. Mapping real time to a discrete time with infinitesimal step size allowed us to cast both (DAE-based) continuous dynamics, and the restart conditions at mode changes, to a uniform framework. This allowed us to properly solve the conflicts mentioned above, between the dynamics of the previous mode and the consistency conditions implied by the new mode. To make all of this doable, we developed a toolbox of new results on nonstandard analysis, mainly related to the needs

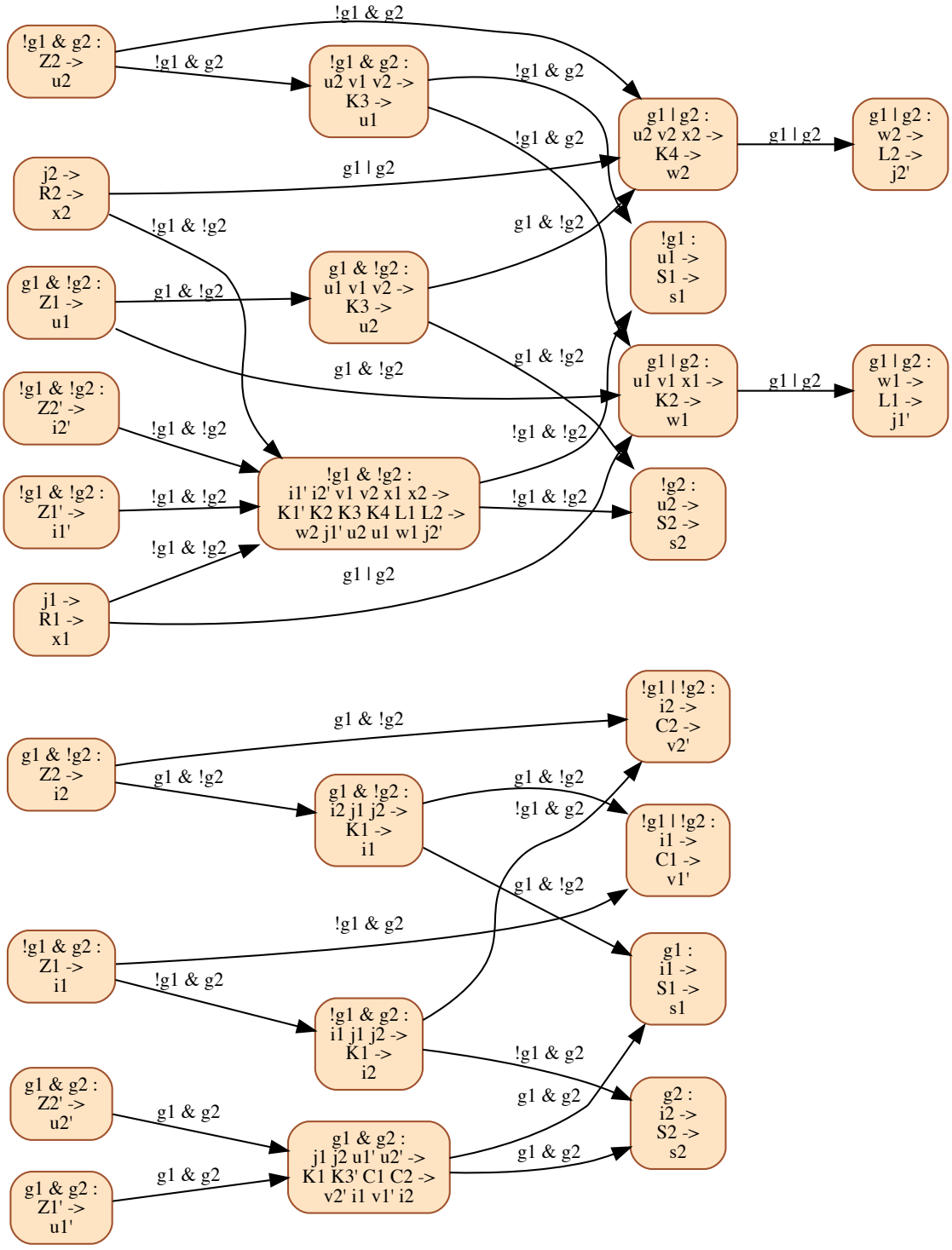


Figure 35: Block dependency graph of the RLDC2 model, generated by IsamDAE. Vertices are labeled $p : R \to B \to W$, where: p is a propositional formula defining in which modes the block is evaluated; R is the set of variables to read; B is the set of equations of the block; W is the set of variables to write. Edges are labeled by a propositional formula, defining in which modes the dependency applies—notation "!g" means "not g".

of structural analysis. We proved that the result of our compilation remains independent from the particular scheme we used to map derivatives to the nonstandard domain.

From our experience in using nonstandard analysis, we can state that this is an extremely convenient tool to establish the mathematical foundations for the compilation of continuous-time systems modeling tools, for both the Simulink and Modelica classes.

A desirable objective of a work like ours is to prove that every execution scheme produced by the approach actually generates solutions of the given source mDAE system. As a reference, such soundness analyses were performed for all synchronous languages [12, 9, 10]. However, a particular difficulty holds: no mathematical definition of solutions of an mDAE system exists in full generality, that would serve as a reference. So far, we were only able to prove such a correctness result for the subclass of "semi-linear multimode systems", that may involve impulsive behaviors at mode changes.

In this work, we paid no specific attention to the efficiency of our structural analysis algorithm, and did not discuss how to generate scalable simulation code. We also only considered flat guards, whereas nested guards (naturally occurring, for instance, in the case of nested "if-then-else" constructs) are very useful in a practical formalism: flattening a model with nested guards is not desirable for getting efficient code. Work is in progress to address these issues by relying on the technique of Binary Decision Diagrams (BDD), and first results are reported in [19], where our IsamDAE tool under development is described.

Acknowledgements

The authors wish to thank a number of people for this work. First, Hilding Elmqvist and Martin Otter introduced them to the subject in the early 2010's; our collaboration then established for a couple of years, leading to both a first version of our approach and the handling of semi-linear mDAE systems; since then, Hilding and Martin have focused on their grand child of Modelica called Modia. We had extensive fruitful discussions with John Pryce on both structural analysis of DAEs, including the Σ-method}, and physical modeling in general; this helped us bring out the need for an explicit distinction between long and transient modes. Khalil Ghorbal participated to the first version of this approach and further interacted with the authors for this revised approach. Finally, Vincent Acary introduced us to nonsmooth systems techniques and solvers and we had frequent exchanges regarding the notions of solution for mDAE systems, revealing to us how strange these can be.

A Appendix: constructive execution of an instant in systems having only long modes

In this appendix, we briefly develop the execution of a run, for mDAE systems according to Definition 27, i.e., without delaying the effect of guards by one nonstandard instant. Since the effect of guards may not be postponed, the current mode is not known from evaluating the previous instant. We still refuse to solve mixed logico-numerical fixpoint equations. Hence we must instead identify the current mode progressively and constructively, according to an iterative scheme of the following form:

1. Enter the new nonstandard instant; the status of a subset of the S-variables is known from previous instant;
2. Using the subset of evaluated guards, select the corresponding set of enabled equations and submit them to index reduction, conflict solving, and then evaluate them;
3. This yields more enabled guards; evaluate them and return to Step 2.

This iteration is performed until, either all S-variables are evaluated to T, F, or I, or no additional guard gets enabled at Step 3—a failure. Since knowing if the current mode is long or transient modifies the way index reduction is performed, and since knowing this requires having evaluated all the guards, we are unable to process general mDAE systems, and we must restrict ourselves to systems having only long modes.

To execute a nonstandard instant through a sequence of microsteps, the algorithm ExecRunProgress (Algorithm 15) is iterated until a failure is reported or a Tick atomic action occurs. In case of success, the pair (σ, Δ) produced by Tick produces the initial status and the context for starting the next instant. ExecRunProgress requires a finite coherent status σ and a finite context Δ; it returns

Algorithm 15 ExecRunProgress; compare with Algorithm 10

Require: (σ, Δ); **return** (*Fail*, updated (σ, Δ))
1: **if** $Success(\sigma)$ **then**
2: $\quad(\sigma, \Delta) \leftarrow \mathsf{Tick}(\sigma)$
3: $\quad\Phi \leftarrow \{e \in \Delta \mid \sigma(\gamma(e)) = \mathrm{T}\}$
4: **else**
5: $\quad G \leftarrow [Enab(\sigma) \cap Undef(\sigma)]$
6: $\quad(\sigma, b, G_\Sigma, \overline{G}_\Sigma) \leftarrow \mathsf{IndexReduc}(G, \sigma)$; increase σ
7: $\quad$**if** $b = \mathrm{F}$ **then return** $Fail(\sigma)$
8: $\quad$**else**
9: $\quad\quad(\sigma, b, H) \leftarrow \mathsf{SolveConflict}((G_\Sigma \cup \overline{G}_\Sigma) \setminus \Phi, \sigma)$; increase σ
10: $\quad\quad$**if** b=F **then return** $Fail(\sigma)$
11: $\quad\quad$**else** $(\sigma, \Phi) \leftarrow \mathsf{Eval}(\Delta, \Phi, \sigma, H)$; increase σ

updated σ and Δ, or documented failure information. Its details are commented next.

Line 1 Function $Success(\sigma)$ checks if status σ is complete according to Definition 39.

Line 2 If σ is complete, then Tick (Algorithm 7) is performed and the instant is completed.

Line 3 Not all the equations belonging to the context are active in the current instant. A system Φ collecting all the *facts* is introduced and is initialized to the subset of the context Δ consisting of all its enabled equations, taking into account the guards that are already known to be true at initialization of the run.

Line 5 The system G is set to the enabled guarded equations not evaluated yet, in status σ.

Line 6 We submit G to the index reduction (IndexReduc atomic action, Algorithm 8). This returns,

- either $b = \text{F}$ (Line 7), in which case ExecRun returns $\mathit{Fail}(\sigma)$ and stops, or
- (Line 8) a pair consisting of G_Σ, a structurally nonsingular system determining some leading variables, and the system $\overline{G}_\Sigma$ collecting consistency equations. In this case, the algorithm can further progress.

Line 9 With the additional help of the set Φ of facts, the possible conflict is solved using the SolveConflict atomic action (Algorithm 9).

Line 10 SolveConflict fails if no structurally regular system survives as a result of applying Line 9. In this case, ExecRun returns $\mathit{Fail}(\sigma)$ and stops.

Line 11 Otherwise, SolveConflict returns a structurally regular system H, which is solved using Eval (82), which updates the status σ by giving a value to more S-variables. As a result, the set Φ of facts is updated.

References

[1] Acary Vincent and Brogliato Bernard. *Numerical Methods for Nonsmooth Dynamical Systems. Applications in Mechanics and Electronics*, volume 35 of *Lecture Notes in Applied and Computational Mechanics*. Springer-Verlag, 2008.

[2] R. Alur, C. Courcoubetis, T. Henzinger, and P. Ho. Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. In *Hybrid systems*, volume 736 of *Lecture Notes in Computer Science*, pages 209–229. Springer Verlag, 1993.

[3] U. M. Ascher and L. R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1st edition, 1998.

[4] A. Benveniste, T. Bourke, B. Caillaud, B. Pagano, and M. Pouzet. A Type-based Analysis of Causality Loops in Hybrid Systems Modelers. *Nonlinear Analysis: Hybrid Systems*, 26:168–189, Nov. 2017.

[5] A. Benveniste, T. Bourke, B. Caillaud, and M. Pouzet. Nonstandard semantics of hybrid systems modelers. *J. Comput. Syst. Sci.*, 78(3):877–910, 2012.

[6] A. Benveniste, B. Caillaud, H. Elmqvist, K. Ghorbal, M. Otter, and M. Pouzet. Structural Analysis of Multi-Mode DAE Systems. Research Report RR-8933, Inria, July 2016.

[7] A. Benveniste, B. Caillaud, H. Elmqvist, K. Ghorbal, M. Otter, and M. Pouzet. Structural analysis of multi-mode DAE systems. In *HSCC*, pages 253–263. ACM, 2017.

[8] 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.

[9] A. Benveniste, B. Caillaud, and P. L. Guernic. Compositionality in dataflow synchronous languages: Specification and distributed code generation. *Inf. Comput.*, 163(1):125–171, 2000.

[10] A. Benveniste, P. Caspi, S. A. Edwards, N. Halbwachs, P. L. Guernic, and R. de Simone. The synchronous languages 12 years later. *Proceedings of the IEEE*, 91(1):64–83, 2003.

[11] C. Berge. *The theory of graphs and its applications*. Wiley, 1962.

[12] G. Berry. Constructive semantics of Esterel: From theory to practice (abstract). In *AMAST '96: Proceedings of the 5th International Conference on Algebraic Methodology and Software Technology*, page 225, London, UK, 1996. Springer-Verlag.

[13] S. Bliudze. *Un cadre formel pour l'étude des systèmes industriels complexes: un exemple basé sur l'infrastructure de l'UMTS*. PhD thesis, Ecole Polytechnique, 2006.

[14] S. Bliudze and D. Krob. Modelling of complex systems: Systems as dataflow machines. *Fundam. Inform.*, 91(2):251–274, 2009.

[15] T. Bourke and M. Pouzet. Zélus: A synchronous language with ODEs. In *Hybrid Systems: Computation and Control (HSCC)*, pages 113–118, Philadelphia, USA, Apr. 2013. ACM.

[16] K. E. Brenan, S. L. Campbell, and L. R. Petzold. *Numerical Solution of Initial Value Problems in Differential-Algebraic Equations*. SIAM, 1996.

[17] D. Broman and J. G. Siek. Modelyze: a gradually typed host language for embedding equation-based modeling languages. Technical Report UCB/EECS-2012-173, EECS Department, University of California, Berkeley, Jun 2012.

[18] R. E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, aug 1986.

[19] B. Caillaud, M. Malandain, and J. Thibault. Implicit structural analysis of multimode DAE systems. In *23rd ACM International Conference on Hybrid Systems: Computation and Control (HSCC 2020)*, Sydney, Australia, April 2020. to appear.

[20] B. Caillaud, M. Malandain, and J. Thibault. Implicit Structural Analysis of Multimode DAE Systems. Research Report RR-9320, Inria Rennes - Bretagne Atlantique ; IRISA, Université de Rennes, Feb. 2020.

[21] S. L. Campbell and C. W. Gear. The index of general nonlinear DAEs. *Numer. Math.*, 72:173–196, 1995.

[22] H. Cartan. *Formes Différentielles*. Collection Méthodes. Hermann, 1967.

[23] N. Cutland. *Nonstandard analysis and its applications*. Cambridge Univ. Press, 1988.

[24] J. Dieudonné. *Fondements de l'analyse moderne*. Gauthier-Villars, 1965.

[25] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices*. Numerical Mathematics and Scientific Computation. Oxford University Press, 1986.

[26] N. Dunford and J. Schwartz. *Linear Operators, Part I, General Theory*. Wiley-Interscience, 1958.

[27] H. Elmqvist, T. Henningsson, and M. Otter. Systems modeling and programming in a unified environment based on Julia. In T. Margaria and B. Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications*, pages 198–217, Cham, 2016. Springer.

[28] H. Elmqvist, S.-E. Mattsson, and M. Otter. Modelica extensions for multi-mode DAE systems. In H. Tummescheit and K.-E. Arzen, editors, *Proc. of the 10th Int. Modelica Conference*, Lund, Sweden, Sept. 2014. Modelica Association.

[29] I. Hasuo. Metamathematics for systems design - comprehensive transfer of formal methods techniques to cyber-physical systems. *New Generation Comput.*, 35(3):271–305, 2017.

[30] W. P. M. H. Heemels, M. K. Camlibel, and J. M. Schumacher. On the dynamic analysis of piecewise-linear networks. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 49(3):315–327, Mar 2002.

[31] 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. SCS/ ACM, 2014.

[32] C. Höger. Elaborate control: variable-structure modeling from an operational perspective. In D. Zimmer and B. Bachmann, editors, *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. ACM, 2017.

[33] Y. Iwasaki, A. Farquhar, V. Saraswat, D. Bobrow, and V. Gupta. Modeling time in hybrid systems: How fast is "instantaneous"? In *IJCAI*, pages 1773–1781, 1995.

[34] D. Liberzon and S. Trenn. Switched nonlinear differential algebraic equations: Solution theory, lyapunov functions, and stability. *Automatica*, 48(5):954–963, 2012.

[35] T. Lindstrøm. An invitation to nonstandard analysis. In N. Cutland, editor, *Nonstandard Analysis and its Applications*, pages 1–105. Cambridge Univ. Press, 1988.

[36] S.-E. Mattsson, M. Otter, and H. Elmqvist. Multi-Mode DAE Systems with Varying Index. In H. Elmqvist and P. Fritzson, editors, *Proc. of the 11th Int. Modelica Conference*, Versailles, France, Sept. 2015. Modelica Association.

[37] S. E. Mattsson and G. Söderlind. Index reduction in Differential-Algebraic Equations using dummy derivatives. *Siam J. Sci. Comput.*, 14(3):677–692, 1993.

[38] V. Mehrmann and L. Wunderlich. Hybrid systems of differential-algebraic equations – analysis and numerical solution. *Journal of Process Control*, 19(8):1218 – 1228, 2009. Special Section on Hybrid Systems: Modeling, Simulation and Optimization.

[39] H. Nakamura, K. Kojima, K. Suenaga, and A. Igarashi. A nonstandard functional programming language. In B. E. Chang, editor, *Programming Languages and Systems - 15th Asian Symposium, APLAS 2017, Suzhou, China, November 27-29, 2017, Proceedings*, volume 10695 of *Lecture Notes in Computer Science*, pages 514–533. Springer, 2017.

[40] C. Pantelides. The consistent initialization of differential-algebraic systems. *SIAM J. Sci. Stat. Comput.*, 9(2):213–231, 1988.

[41] P. Pepper, A. Mehlhase, C. Höger, and L. Scholz. A compositional semantics for modelica-style variable-structure modeling. In *th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, 2011.

[42] L. Petzold. Differential algebraic equations are not ODEs. *SIAM J. Sci. Stat. Comput.*, 3:367–384, 1982.

[43] F. Pfeiffer. On non-smooth multibody dynamics. *Proceedings of the Institution of Mechanical Engineers, Part K: Journal of Multi-body Dynamics*, 226(2):147–177, 2012.

[44] F. Pfeiffer and C. Glocker. *Multibody Dynamics with Unilateral Contacts*. Wiley, 2008.

[45] A. Platzer. *Logical Foundations of Cyber-Physical Systems*. Springer Publishing Company, Incorporated, 1st edition, 2018.

[46] A. Pothen and C. Fan. Computing the block triangular form of a sparse matrix. *ACM Trans. Math. Softw.*, 16(4):303–324, 1990.

[47] D. Potop-Butucaru, S. A. Edwards, and G. Berry. *Compiling Esterel*. Springer, 2007.

[48] J. D. Pryce. A simple structural analysis method for DAEs. *BIT*, 41(2):364–394, 2001.

[49] A. Robinson. *Nonstandard Analysis*. Princeton Landmarks in Mathematics, 1996. ISBN 0-691-04490-2.

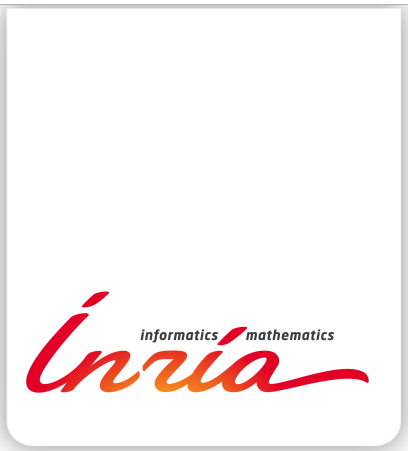
[50] S. Schoeder, H. Ulbrich, and T. Schindler. Discussion of the Gear–Gupta–Leimkuhler method for impacting mechanical systems. In *Multibody System Dynamics*, volume 31, pages 477–495, 2013.

[51] K. Suenaga, H. Sekine, and I. Hasuo. Hyperstream processing systems: nonstandard modeling of continuous-time signals. In R. Giacobazzi and R. Cousot, editors, *The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '13, Rome, Italy - January 23 - 25, 2013*, pages 417–430. ACM, 2013.

[52] S. Trenn. *Distributional Differential Algebraic Equations*. PhD thesis, Technischen Universität Ilmenau, 2009.

[53] S. Trenn. Regularity of distributional differential algebraic equations. *MCSS*, 21(3):229–264, 2009.

[54] D. Zimmer. *Equation-Based Modeling of Variable-Structure Systems*. PhD thesis, ETH Zürich, No. 18924, 2010.



RESEARCH CENTRE
RENNES – BRETAGNE ATLANTIQUE

Campus universitaire de Beaulieu
35042 Rennes Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399