Structural Analysis of Multi-Mode DAE Systems

Albert Benveniste, Benoît Caillaud, Hilding Elmqvist, Khalil Ghorbal, Martin Otter, Marc Pouzet

Structural Analysis of Multi-Mode DAE Systems

Albert Benveniste , Benoît Caillaud , Hilding Elmqvist , Khalil Ghorbal , Martin Otter , Marc Pouzet

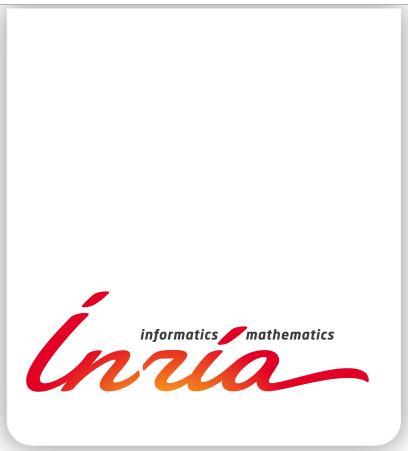
RESEARCH REPORT

N° 8933

July 2016

Project-Teams Hycomes, Parkas

ISSN 0249-6399 ISRN INRIA/RR--8933--FR+ENG



Structural Analysis of Multi-Mode DAE Systems

Albert Benveniste [*], Benoît Caillaud [*], Hilding Elmqvist [†], Khalil Ghorbal [*] [‡], Martin Otter [§], Marc Pouzet [¶]

Project-Teams Hycomes, Parkas

Research Report n° 8933 — version 3 — initial version July 2016 — revised version June 2017 — 23 pages

Abstract: Differential Algebraic Equation (DAE) systems constitute the mathematical model supporting physical modeling languages such as Modelica, VHDL-AMS, or Simscape. Unlike ODEs, they exhibit subtle issues because of their implicit *latent equations* and related *differentiation index*. Multi-mode DAE (mDAE) systems are much harder to deal with, not only because of their mode-dependent dynamics, but essentially because of the events and resets occurring at mode transitions. Unfortunately, the large literature devoted to the numerical analysis of DAEs does not cover the multi-mode case. It typically says nothing about mode changes. This lack of foundations cause 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 to the *structural analysis* of mDAE systems which properly extends the usual analysis of DAE systems. We define a constructive semantics based on nonstandard analysis and show how to produce execution schemes in a systematic way.

This report is an extended version of the publication [2].

Key-words: Multi-mode systems, differential algebraic equations, DAE, differential index, structural analysis, operational semantics, nonstandard analysis

This work is based on research performed within the ITEA2 project MODRIO with partial financial support from the French DGE (Direction Générale des Entreprises), Swedish VINNOVA (Stärker Sveriges innovationskraft för hållbar tillväxt och samhällsnytta) and German BMBF (Bundesministerium für Bildung und Forschung).

[*] INRIA, Rennes, France
[†] Mogram AB, Lund, Sweden
[‡] Corresponding author: khalil.ghorbal@inria.fr
[§] DLR Institute of System Dynamics and Control, Oberpfaffenhofen, Germany
[¶] Ecole Normale Supérieure (ENS), Paris

RESEARCH CENTRE
RENNES – BRETAGNE ATLANTIQUE

Campus universitaire de Beaulieu
35042 Rennes Cedex

Analyse Structurelle des systèmes de DAE multi-modes

Résumé : La modélisation des systèmes physiques s'effectue de plus en plus à l'aide de langages utilisant des DAE (Differential Algebraic Equation), et non plus seulement des ODE (Ordinary Differential Equation). L'exemple le plus connu est Modelica, mais il existe d'autres outils (VHDL-AMS, Simscape, en particulier). Les DAE présentent une difficulté nouvelle par rapport aux ODE: les notions d'*équation latente* et d'*index de différentiation*. Ces deux notions ont été proposées et étudiées en profondeur par les mathématiciens depuis bientôt trente ans et jouent un rôle important dans les outils de modélisation, à travers ce qu'on appelle l'*analyse structurelle* de ces systèmes.

En revanche, les systèmes de DAE *multi-modes* (où la dynamique dépend du mode où se trouve le système) sont beaucoup plus difficiles à étudier; non seulement parce que la dynamique dépend du mode, mais surtout à cause des événements de changement de mode, avec les réinitialisations associées. Et, de fait, les systèmes de DAE multi-modes ont été peu étudiés en profondeur. Avec pour conséquence que le traitement des changements de mode est insuffisamment compris, et que les outils de modélisation ne traitent que des classes restreintes de modèles, avec une absence de définition claire de la nature de ces restrictions.

Dans ce travail, nous présentons une approche systématique, mathématiquement fondée, pour l'analyse structurelle des systèmes de DAE multi-modes. Nous utilisons l'analyse non-standard pour ramener l'analyse structurelle à un problème sur des systèmes dynamiques à temps discret, et nous réutilisons les idées provenant de la *sémantique constructive* des langages synchrones. Nous complétons ceci par des techniques de standardisation (provenant de l'analyse non-standard) pour obtenir le code éxécutable final, où une structure d'automate coiffe et coordonne le travail des solveurs.

Ce rapport de recherche est une version augmentée de la publication [2].

Mots-clés : Systèmes multi-mode, équations différentielles algébriques, DAE, indice différentiel, analyse structurelle, sémantiques opérationnelles, analyse non standard

Contents

1 Introduction

Multi-mode DAE systems constitute the mathematical model supporting physical modeling languages such as Modelica. Multi-mode DAE models can be represented as systems of equations of the form

$$\begin{array}{ll} \text{if} & \gamma_j(\text{the } x_i \text{ and their derivatives}) \\ \text{do} & f_j(\text{the } x_i \text{ and their derivatives}) = 0 \end{array} \quad (1)$$

where $x_1, \ldots, x_n$ denote the system variables, $\gamma_j(\ldots)$ is a predicate guarding the DAE $f_j(\ldots) = 0$. The meaning is that, if γ_j has the value true, then equation $f_j(\ldots) = 0$ has to hold, otherwise it is discarded. In particular, when all the predicates are the constant true, one obtains a *single-mode* DAE, that is a standard DAE defined by the set of equations $f_j(\ldots) = 0$. When the functions f_j have the special form $x'_j - g_j(x_1, \ldots, x_n)$, one recovers the usual Ordinary Differential Equations (ODE) system $x'_j = g_j(x_1, \ldots, x_n)$. 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 $x'_1, \ldots, x'_n$. Finally, our modeling framework is fully compositional, since systems of systems of equations of the form (1) are just systems of equations (with eventually additional constraints connecting the different state variables).

Solving numerically single-mode DAEs faces the well known issue of *differentiation index* [6], originating from the possible existence of so-called *latent constraints*. Informally, latent constraints in DAE systems are additional equations obtained from the original equations $f_j(\ldots) = 0$ by time differentiation, assuming the existence of smooth enough solutions for those extra equations to be well-defined. A DAE has differential index n if one or more equations must be differentiated n-times until the equations can be algebraically transformed to an ODE form with the x_i as states. In particular, ODEs are fully explicit differential equations and are therefore DAEs of index 0. In practice, systems with index greater than 1 are common (e.g., the DAE of a pendulum in Cartesian coordinates has index 3) and higher indexes are often encountered in common Modelica models. The *Structural Analysis* of DAE systems, such as the Pantelides algorithm [14], is an abstract lightweight graph-based analysis that constructively computes a "structural" differentiation index which can be formally related to the numerical differentiation index. Such structural analysis is often performed as a pre-processing step before calling numerical solvers.

Unlike single-mode DAE systems, however, no theory exists that supports the structural analysis of multi-mode DAE systems. The usual approach consists in performing the structural analysis for each mode. This, however, tells nothing about how mode changes could be handled. Even more so when mode changes occur in cascades.

Related Work: Multi-domain modeling languages that support DAEs such as Modelica or VHDL-AMS, but also proprietary languages such as Simscape have typically the restriction that the number of equations cannot change during simulation. Modeling tools have further restrictions, e.g. that the DAE index cannot change during simulation, or that impulses occurring due to mode switches are not supported. There are some proposals such as [11] that try to handle multi-mode DAEs by using source to source model transformations to bring the model in a form that is amenable to known structural analysis and index reduction techniques. The class of supported models is still however restricted, e.g., mode changes leading to impulses cannot be handled. On the other hand, there is a long tradition for mechanical systems to handle contact problems and friction which lead to mode changes, index changes and/or impulses. An overview of the actual state of the art is for example given in [15]. It is, however, not obvious how this domain-specific approach can be generalized.

To our knowledge, the only work addressing the structural analysis of multi-mode DAE systems is [13]. While this work contains interesting results regarding numerical techniques to detect chattering between modes, it assumes deterministic multi-mode systems where consistent resets are already known for each mode. Such assumptions do not hold in general, especially for a compositional framework where one wants to assemble pre-defined physical components. Besides, for complex systems, one often resorts to simulations to better understand resets and mode changes. In this work, we attempt to constructively build deterministic

and causal execution schemes. In a sense, our analysis could be regarded as a pre-processing step to perform prior to simulating multi-mode DAE systems.

Contributions: In this paper, we consider systems of equations of the form (1) as a core framework for multi-mode DAE systems. This modeling framework is fully equational and compositional. We define a constructive (small-step) semantics for such framework by relying on nonstandard analysis [10, 1]. We handle in a unified way, discrete and possibly impulsive mode changes on one hand, and purely continuous evolution within one mode on the other hand. This makes it possible to formally define which systems a compiler should accept/refuse. We finally explain how to generate an execution scheme from the nonstandard constructive semantics. We illustrate the different steps of our analysis on a simple, yet challenging, example we explain next.

2 A Simple Clutch

We consider a simple, idealized clutch involving two rotating shafts where no motor or brake are connected. The dynamics of each shaft i is modeled by $\omega_i' = f_i(\omega_i, \tau_i)$ for some functions f_i, where ω_i is the angular velocity, τ_i is the torque applied to the 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 ($\gamma = \mathrm{T}$) or released ($\gamma = \mathrm{F}$). When the clutch is released, the two shafts rotate independently: no torque is applied ($\tau_1 = \tau_2 = 0$). 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 ($\tau_1 + \tau_2 = 0$). If the clutch is initially released, then at the instant of contact the relative speed of the two rotating shafts jumps to zero and, as a consequence, an impulse generally occurs on the torques. This idealized clutch model is not supported by the existing Modelica tools at the date of this writing—we later give explanations about what the difficulty is. The clutch model is summarized below.

$$\left\{\begin{array}{lll} & & \omega_1' = f_1(\omega_1, \tau_1) \quad (e_1) \\ & & \omega_2' = f_2(\omega_2, \tau_2) \quad (e_2) \\ \texttt{if } \gamma & \texttt{do} & \omega_1 - \omega_2 = 0 \quad (e_3) \\ & \texttt{and} & \tau_1 + \tau_2 = 0 \quad (e_4) \\ \texttt{if not } \gamma & \texttt{do} & \tau_1 = 0 \quad (e_5) \\ & \texttt{and} & \tau_2 = 0 \quad (e_6) \end{array}\right. \tag{2}$$

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 Section 2.4. For convenience, we recall basic notions of nonstandard analysis in Section 2.3.

2.1 Separate Analysis of Each Mode

In the released mode, when γ is false in System (2), the two shafts are independent and one obtains the following two independent ODEs for ω_1 and ω_2:

$$\begin{array}{llll} \omega_1' = f_1(\omega_1, \tau_1) & (e_1) & \tau_1 = 0 & (e_5) \\ \omega_2' = f_2(\omega_2, \tau_2) & (e_2) & \tau_2 = 0 & (e_6) \end{array} \tag{3}$$

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

$$\begin{array}{llll} \omega_1' = f_1(\omega_1, \tau_1) & (e_1) & \omega_1 - \omega_2 = 0 & (e_3) \\ \omega_2' = f_2(\omega_2, \tau_2) & (e_2) & \tau_1 + \tau_2 = 0 & (e_4) \end{array} \tag{4}$$

Due to the additional constraints (e_3) and (e_4), System (4) is no longer an ODE, but rather a DAE. Notice in particular that the derivatives of the torques are not explicitly given and that the state variables ω_i have to satisfy the extra constraint (e_3) as long as the system evolves in that mode.

If one is able to uniquely determine the so called leading variables $(\omega_1', \omega_2', \tau_1, \tau_2)$ given a *consistent* value for the *state variables* (ω_1, ω_2), one could regard the DAE as an "extended ODE" [17] where an integration step is performed to update the current positions (ω_1, ω_2) using the computed values for their derivatives (ω_1', ω_2'). Here, by consistent values for (ω_1, ω_2) we mean a pair that satisfies (e_3).

It turns out that this does not work for System (4) as is. To intuitively explain what the problem is, we

move to discrete time by applying an explicit first order Euler scheme with constant step size $\delta > 0$:

$$\begin{array}{llll} \omega_1^\bullet = \omega_1 + \delta \cdot f_1(\omega_1, \tau_1) & (e_1^\delta) & \omega_1 - \omega_2 = 0 & (e_3) \\ \omega_2^\bullet = \omega_2 + \delta \cdot f_2(\omega_2, \tau_2) & (e_2^\delta) & \tau_1 + \tau_2 = 0 & (e_4) \end{array} \tag{5}$$

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 initial values for ω_1 and ω_2, i.e., satisfying (e_3). Attempting to apply the Euler scheme (5) fails in that, generically, there is no unique values for the $\omega_i^\bullet$. Indeed, we have only three equations e_1^δ,e_2^δ, and e_4 for four unknowns, τ_1, τ_2, $\omega_1^\bullet$, and $\omega_2^\bullet$. However, since System (5) is time invariant, and assuming that the system remains in the engaged mode for at least δ seconds, there exists an additional *latent constraint* on the set of variables $(\omega_1, \omega_2, \tau_1, \tau_2, \omega_1^\bullet, \omega_2^\bullet)$, namely

$$\omega_1^\bullet - \omega_2^\bullet = 0 \qquad (e_3^\bullet) \tag{6}$$

obtained by shifting (e_3) forward. One can now use System (5) augmented with Eq. (6) to get an execution scheme for the engaged mode of the clutch (see Exec. Sch. 1 below).

Execution Scheme 1 System (5)+Eq. (6).

Require: consistent ω_1 and ω_2, i.e., satisfying (e_3).
1: `Solve` $\{e_1^\delta, e_2^\delta, e_3^\bullet, e_4\}$ ▷ 4 equations, 4 unknowns
2: $(\omega_1, \omega_2) \leftarrow (\omega_1^\bullet, \omega_2^\bullet)$ ▷ Update (ω_1, ω_2)
3: `Tick` ▷ Move to next discrete step

Since the new values of the state variables satisfy (6) by construction, the consistency condition is met at the next iteration step (should the system remains in the same mode). 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,[1] because we are solving four algebraic equations involving four dependent variables.

Observe that the same analysis could be applied to the original continuous time dynamics (System (4)) by augmenting the latter with the following *latent differential equation*:

$$\omega_1' - \omega_2' = 0 \qquad (e_3') \tag{7}$$

obtained by differentiating (e_3)—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 (compare with Exec. Sch. 1).

Execution Scheme 2 System (4)+Eq. (7).

Require: consistent ω_1 and ω_2, i.e., satisfying (e_3).
1: `Solve` $\{e_1, e_2, e_3', e_4\}$ ▷ 4 equations, 4 unknowns
2: `ODESolve` (ω_1, ω_2) ▷ Update (ω_1, ω_2)
3: `Tick` ▷ Move to next 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'). In Exec. Sch. 1, getting the next values for the ω_1 and ω_2 was straightforward. In Exec. Sch. 2, however, the derivatives (ω_1', ω_2') are first evaluated, and then used to update the state by using an ODE solver (here denoted by `ODESolve`). Note that, when considering an exact mathematical solution, if $\omega_1 - \omega_2 = 0$ holds initially and $\omega_1' - \omega_2' = 0$, then the linear constraint (e_3) will be satisfied for all positive time.

Exec. Sch. 2 is known in the literature as the method of *dummy derivatives* [12]. It requires adding the (smallest set of) latent equations needed for Line 1 of the execution scheme to become solvable and deterministic. The maximal amount of successive differentiation operations needed in obtaining all the latent equations is called the *differentiation index* [6], or simply the *index*. In Exec. Sch. 2, differentiating (e_3) once was sufficient. If, e.g., the second derivative of the state variables were involved in the system model, then, two successive differentiations would be needed. Observe that both execution schemes 1 and 2 rely on an algebraic equation system solver.

To conclude this section, we briefly discuss the initialization problem. Unlike ODE systems, the initialization problem is far from trivial for DAE systems, even more so when the state variables have to satisfy additional user-defined constraints. This is in fact often the case for multi-mode systems since the

[1] See Section 3.1 for what is formally meant by "structural" in this context.

system has to start a new mode from a previously known state. For the clutch example, if one considers System (4) as a standalone DAE, the initialization is performed as indicated in Exec. Sch. 3.

Execution Scheme 3 Initialization of System (4)+Eq. (6).

1: $(\omega_1, \omega_2, \tau_1, \tau_2, \omega_1', \omega_2') \leftarrow \texttt{Solve}\{e_1, e_2, e_3, e_3', e_4\}$
▷ 5 equations, 6 unknowns

Note that we have 6 unknowns and only 5 equations, so we are left with 1 degree of freedom—mathematically speaking, the set of all initial values for the 6-tuple of variables is a manifold of dimension 1. For example, one can freely fix the initial common rotation speed so that (e_3) is satisfied. Notice that the latent equation (e_3') is mandatory in order to determine the initial value of the torques τ_i.

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 reduces to applying the initialization given in Exec. Sch. 3. If one was to treat resets at mode changes as initializations, it would mean that the clutch system is nondeterministic precisely because of the *extra* degree of freedom of Exec. Sch. 3. In contrast, the physics tells us that the state of the system should be entirely determined when the clutch gets engaged. This, therefore, comforts the intuition that resets at mode changes are not mere initializations.

If, however, one considers the known values of the state variables "immediately" before switching to the engaged mode, the system becomes over-determined as generically the equation (e_3) won't be satisfied. In this case, it is unclear what constraint should be relaxed and why.

This is precisely why this clutch model cannot be simulated as is with Modelica tools. A work around would be to compute and specify reset values by hand in the model. Such approach, however, impairs modularity since significant additional manual work is needed when building the clutch model from the two separate models for each mode.

We present next our approach to tackle such problems using nonstandard analysis.

2.3 Nonstandard Semantics

While the meaning of the clutch model in System (2) is fully clear when the system evolves continuously inside one of the two modes, the model does not say explicitly what happens at mode changes. We are in particular interested in two specific issues:

- (i) in case of discontinuous trajectories, what meaning one can give to the equations involving derivatives and what role those equations play in determining the discontinuity gap.
- (ii) if an event enables new constraints that make the system overdetermined, then what constraints one has to relax (and why) for the simulation to proceed.

To answer those questions, we use the *nonstandard analysis* [10] and in particular the nonstandard semantics of hybrid systems introduced in [1]. Nonstandard reals, a.k.a. hyperreals, denoted by ${}^\star\mathbb{R}$, extend the usual reals with *infinitesimals* and *infinite* numbers. A totally ordered field, ${}^\star\mathbb{R}$, is constructed from the reals very much like $\mathbb{R}$ is constructed from the rationals using Cauchy sequences. Following the construction proposed by [10], ${}^\star\mathbb{R}$ is defined as the set of all (not necessarily converging) sequences of real numbers $x = \{x_n\}$ quotiented by some equivalence relation $\sim$ such that, if the sequences x and y only differ by a finite number of elements, then $x \sim y$ holds, see [10] for details. Any real number $a \in \mathbb{R}$ is lifted to a hyperreal ${}^\star a = \{a, a, a, \dots\}$. A hyperreal ε is said to be infinitesimal if $|\varepsilon| < r$ for all positive real numbers r. For instance, $\{n^{-1}\}_{n\in\mathbb{N}^*}$ and $\{n^{-2}\}_{n\in\mathbb{N}^*}$ are (positive) infinitesimals. We denote $x \approx y$ if $x - y$ is an infinitesimal ($\approx$ should not be confused with the relation $\sim$ used in the construction of ${}^\star\mathbb{R}$). Any finite hyperreal x possesses a unique real number $\mathrm{st}(x)$ such that $x \approx \mathrm{st}(x)$, we call it its *standard part* [10].

Functions over the reals can be *internalized* as functions over the hyperreals by considering the constant sequence formed by the same function. If $x : t \mapsto x(t)$

denotes a function defined over $\mathbb{R}$, and $\partial = \{\partial_n\}$ denotes an infinitesimal then one defines ${}^\star x(t+\partial)$ as the infinite sequence formed by $x(t+\partial_n)$. To simplify the notations, we will simply write x instead of ${}^\star x$ whenever the distinction is clear from the context. We now formally define the immediate next value of a function we used earlier for the clutch example—such notion does not exist over the reals.

Definition 1 (Forward Shift) *Let x be a real valued function defined over $[t,s)$ for some $t,s \in \mathbb{R}$, $t<s$. Let ∂ denote a positive infinitesimal. We define $x^\bullet \in {}^\star\mathbb{R}$ as*

$$x^\bullet(t) =_{\text{def}} x(t+\partial) \ .$$

Observe that $t+\partial < s$ for any positive infinitesimal ∂ (by definition of the infinitesimals). Notice also that the definition of the forward shift is dependent on the infinitesimal ∂.

Solutions of multi-mode DAEs may be non differentiable and even non continuous at events of mode change. To give a meaning to $x'(t)$ at a an arbitrary point t of a function $x : t \mapsto x(t)$, we define it as the nonstandard difference quotient of x at t for a fixed positive infinitesimal ∂:

$$x'(t) \ =_{\text{def}} \ \frac{x(t+\partial)-x(t)}{\partial} \ . \tag{8}$$

The definition of x' by equation (8) is motivated by the following characterization of derivatives in nonstandard analysis: a function $f : \mathbb{R} \to \mathbb{R}$ is differentiable at $a \in \mathbb{R}$ if and only if there exists a real number b such that

$$\frac{f(a+\varepsilon)-f(a)}{\varepsilon} \approx b$$

for all non zero infinitesimals ε (See for instance Proposition I.3.5 in [10]), where we recall that $u \approx v$ means that $u-v$ is infinitesimal. Then b is equal to the derivative $f'(a)$. Thus, (8) coincides up to $\approx$ with the derivative of x at the instants when x is differentiable. In a multi-mode DAE system, substituting every occurrence of $x'(t)$, for every $t \in \mathbb{R}$, by its expression in (8) yields a difference algebraic equation (dAE) system.[2]

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

Applying such substitution for System (2) gives the following multi-mode dAE (mdAE):

$$\left\{\begin{array}{lll} & \frac{\omega_1^\bullet-\omega_1}{\partial} = f_1(\omega_1,\tau_1) & (e_1^\partial) \\ & \frac{\omega_2^\bullet-\omega_2}{\partial} = f_2(\omega_2,\tau_2) & (e_2^\partial) \\ \texttt{if } \gamma \ \texttt{do} & \omega_1-\omega_2=0 & (e_3) \\ \texttt{and} & \tau_1+\tau_2=0 & (e_4) \\ \texttt{if not } \gamma \ \texttt{do} & \tau_1=0 & (e_5) \\ \texttt{and} & \tau_2=0 & (e_6) \end{array}\right. \tag{9}$$

The state variables are ω_1,ω_2 whereas the leading variables are now γ,τ_1,τ_2, and $\omega_1^\bullet,\omega_2^\bullet$. Notice that we now add the guard γ to the set of leading variables. The rationale is that γ is an input variable and is not evaluated at the previous instant (unlike the state variables ω_1,ω_2).

Following the reasoning of Section 2.1, one sees at once that within each mode, one obtains 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 (9) with respect to the explicit Euler scheme of Section 2.1 is twofold: first, it is exact up to infinitesimals within each mode, and, second, it will allow us to carefully analyze what happens at events of modes change.

We now move to deducing an execution scheme for System (9). We obey the following Causality Principle: since γ is a guard, it must be evaluated prior the DAE it controls. Then, the following cases occur:

Case 1. If γ=F, equations (e_1^∂), (e_2^∂), (e_5) and (e_6) are enabled and can be evaluated, one at a time, in the following order: (e_5) sets τ_1 to 0; (e_6) sets τ_2 to 0; then (e_1^∂) is solved to compute $\omega_1^\bullet$; and finally (e_2^∂) is solved to compute $\omega_2^\bullet$.

Case 2. If γ=T, equations (e_3) and (e_4) become enabled with the notable difference that (e_3) involves the state variables ω_i, which values were set at the previous instant. (This contrasts with (e_5) and (e_6) in the previous case where only the τ_i are involved.) Two possible subcases follow, depending on the status of the consistency equation (e_3):

Case 2.1. If $\omega_1-\omega_2=0$ holds, then we are left with equations $(e_1^\partial),(e_2^\partial),(e_4)$ with dependent variables $\omega_1^\bullet,\omega_2^\bullet$, τ_1,τ_2, which brings us back to the underdetermined case we discussed about System (5):

we add the latent equation $\omega_1^\bullet - \omega_2^\bullet = 0$. Note that $\omega_1 - \omega_2 = 0$ provably holds if we were already in the same mode at the previous instant. Since $\omega_1 - \omega_2 = 0$ and $\omega_1^\bullet - \omega_2^\bullet = 0$ together imply $\omega_1' - \omega_2' = 0$ by (8), this case gives the nonstandard version of the continuous dynamics (4) augmented with the latent equation (7), within the engaged mode.

Case 2.2. If, however, the previous time step sets values for the state variables such that $\omega_1 - \omega_2 \neq 0$, the system is overdetermined and the consistency equation (e_3) cannot be satisfied. A first idea would be to reject this model. This would be unfortunate as the original (standard) model seemed natural for the clutch. To overcome this issue, we defer the enabled equation (e_3) (which made the system overdetermined) to an immediate next instant $t + \partial$. This amounts to replacing the equation (e_3) by its forward shift $(e_3^\bullet) : \omega_1^\bullet - \omega_2^\bullet = 0$. By doing so, one hopes that the system recovers a consistent initial condition for the new mode in an infinitesimal time step, starting from its previous non consistent state.

The corresponding nonstandard execution scheme is summarized in Exec. Sch. 4. We use the variable Δ to encode the *context*: that is the equations known to be satisfied by the state variables. For instance, $e_3 \notin \Delta$ corresponds to **Case 2.2** above. At each tick, the context gets eventually updated to account for the equations that the new state satisfies. The procedure `Reset` solves the system of equations in its argument to determine the reset values of the state variables (the computation is detailed next in Section 2.4). The procedure `Solve`, solves the (algebraic) system to determine the new values of the leading variables.

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

Execution Scheme 4 for Nonstandard System (9).

Require: ω_1 and ω_2.
1: **if** γ **then**
2: **if** $e_3 \notin \Delta$ **then**
3: $(\omega_1^\bullet, \omega_2^\bullet) \leftarrow$ `Reset` $\{e_1^\partial, e_2^\partial, e_3^\bullet, e_4\}$
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**
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`: $\Delta \leftarrow \Delta \setminus \{e_3\}$

2.4 Standardization

Exec. Sch. 4 cannot be executed as is 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 $\mathrm{st}(x) \in \mathbb{R}$ such that $x \approx \mathrm{st}(x)$. 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 the sojourn time in each mode is not reduced to a single point, and discrete evolutions at events of mode change.

Standardization within continuous modes: If $x : t \mapsto x(t)$, $t \in [s, p)$, denotes the real continuous solution at a given mode, then, such solution is in particular differentiable for all t in $[s, p)$. Thus, for all $t \in [s, p)$, there exists a real number

$$x'(t) \approx \frac{x(t+\partial) - x(t)}{\partial} = \frac{x^\bullet(t) - x(t)}{\partial}\,.$$

In the sequel, for a given non zero infinitesimal ξ, we denote by $o(\xi)$ any nonstandard real number such that $o(\xi)/\xi$ is itself infinitesimal. For instance, any infinitesimal is $o(1)$. Notice that the "o" notation has thus a precise fixed meaning in nonstandard analysis. Hence equations (e_1^∂) and (e_2^∂) rewrite $\omega_i' = f_i(\omega_i, \tau_i) + o(1)$, which can be shown to standardize as the differential equations (e_1) and (e_2), respectively. This suffices to recover the dynamics (3) in the released mode $\gamma = \mathrm{F}$.

In the engaged mode $\gamma = \mathrm{T}$, 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 (4) augmented with the latent equation (7) in the engaged mode $\gamma = \mathrm{T}$.

Standardization 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. However, the equations defining the $\omega_i^\bullet$ as functions of the ω_i involve the hyperreal ∂ as an explicit coefficient. To recover real values for the next state we must standardize the nonstandard expressions of the state variables.

The transition $\gamma : \text{T} \to \text{F}$ raises no particular problem, since the target mode has an ODE dynamics whose state variables are initialized by using the corresponding exit values when leaving the previous mode.

The transition $\gamma : \text{F} \to \text{T}$ 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

$$\frac{\omega_i^\bullet - \omega_i}{\partial} = f_i(\omega_i, \tau_i), \quad i = 1, 2. \tag{10}$$

Assuming $\omega_1 - \omega_2 \neq 0$, since $\omega_1^\bullet - \omega_2^\bullet = 0$ holds, the right difference quotient

$$\frac{(\omega_1^\bullet - \omega_2^\bullet) - (\omega_1 - \omega_2)}{\partial} = f_1(\omega_1, \tau_1) - f_2(\omega_2, \tau_2)$$

cannot be a finite nonstandard real because if it was, that would mean that the function $\omega_1(t) - \omega_2(t)$ is right continuous which contradicts the assumption that $\omega_1 - \omega_2 \neq 0$ at the immediate previous time. Thus, 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 we started at a 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*. This informal "impulse analysis" can be formalized by abstracting variables by their *magnitude order* with respect to the infinitesimal ∂. For instance, the magnitude order of the finite hyperreals is zero, whereas the magnitude order of an infinite (or impulse) of the form $\partial^{-1} r$ for a finite non zero real number r is 1. (See Appendix A.1 for more details about the impulsive analysis).

It remains to compute the (standard) reset values for the state variables. To simplify our exposure, we assume that the f_i are linear in their arguments, i.e., f_i has the following form:

$$f_i(\omega_i, \tau_i) \quad = \quad a_i \omega_i + b_i \tau_i, \tag{11}$$

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 moment of inertia. This yields the following dynamics at the instant when γ switches from F to T:

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

Eliminating the torques τ_i yields

$$\omega_i^\bullet \quad = \quad \frac{b_2\omega_1 + b_1\omega_2}{b_1 + b_2} + \partial.\frac{a_1 b_2 \omega_1 + a_2 b_1 \omega_2}{b_1 + b_2}$$

and therefore, the standard part of $\omega_i^\bullet$ is

$$\text{st}(\omega_i^\bullet) \quad = \quad \frac{b_2\omega_1 + b_1\omega_2}{b_1 + b_2}, \tag{12}$$

that is the weighted arithmetic mean of ω_1 and ω_2. Eq. (12) 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 can be discarded. The above direct rewriting technique is limited to this linear case. We develop in Appendix A.2 a technique that applies whenever Taylor expansions are available for the functions f_i.

As a final observation, instead of computing the exact standard part of $\omega_i^\bullet$, one could instead attempt to approximate it by substituting ∂ with a small (but non infinitesimal) step size δ.It would then be interesting to study more in depth the numerical accuracy and convergence of such schemes by focusing on state variables only and ignoring the impulsive ones. We leave this as a future work.

Figure 1 shows a simulation of the clutch model where the resets are explained above. One can see that the reset value is, as one may expect physically, between the two values of ω_1 and ω_2 when $\gamma : \text{F} \to \text{T}$ (at $t = 5$s), and that the transition is continuous at the second reset (at $t = 10$s).

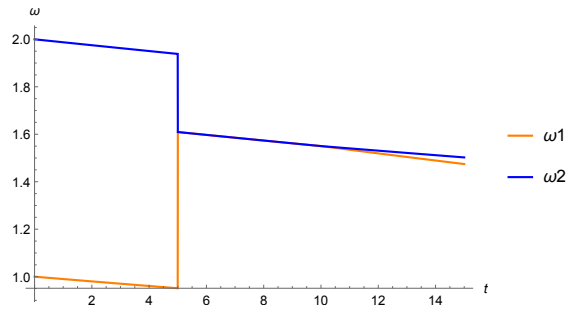


Figure 1: Simulation of the clutch model with resets. Mode change F → T occurs at $t = 5s$ and mode change T → F occurs at $t = 10s$.

3 Structural Analysis

Formally, defining how to derive execution code for a general mDAE system is a challenging problem, because the already difficult structural analysis for DAE systems [6] gets complicated by the need for a structural analysis of mode changes as well. In this section, we propose a novel approach to this problem, based on a formalization of the intuitions developed on the clutch example (Section 2).

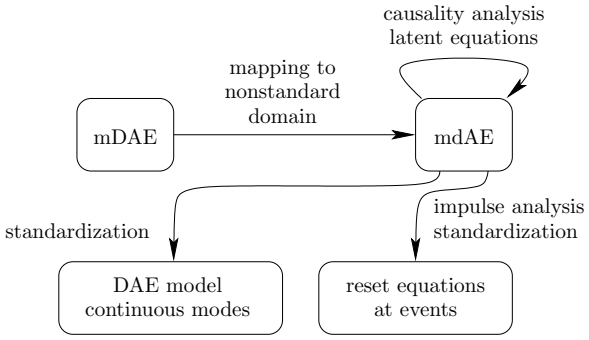


Figure 2: Structural analysis of mDAE systems.

As depicted in Figure 2, our method decomposes into several steps. The first step consists in transforming the mDAE system into a system of multi-mode difference Algebraic Equations (mdAE) using the nonstandard interpretation of the derivatives. The second step applies Algorithm 5 (See Section 3.3) to the mdAE system. The algorithm performs a structural analysis resulting in a new mdAE system where latent equations and a scheduling of blocks of equations are made explicit. The last steps are standardization steps, where the smooth dynamics in each mode, and the possibly discontinuous/impulsive state jumps occurring at mode changes, are recovered from the latter mdAE system.

3.1 Background

As a background and to contrast the differences and the inherent difficulties of mDAEs, we first recall the structural analysis for DAE systems (single-mode) before extending it to the multi-mode case.

Consider a system of smooth algebraic equations with n equations and n dependent variables (unknowns) $y_1, \ldots, y_n$:

$$f_j(x_1, \ldots, x_m, y_1, \ldots, y_n) = 0, \quad j = 1, \ldots, n \quad (13)$$

rewritten as $F(X, Y) = 0$ where X and Y denote the vectors $(x_1, \ldots, x_m)$ and $(y_1, \ldots, y_n)$, respectively, and F is the vector $(f_1, \ldots, f_n)$. The Implicit Function Theorem (see, e.g., Theorem 10.2.2 in [7]) states that, if $(u, v) \in \mathbb{R}^{m+n}$ is a value for the pair (X, Y) such that $F(u, v) = 0$ and the Jacobian of F with respect to Y (denoted by $\nabla_Y F$) at the point (u, v) is nonsingular, then there exists, in an open neighborhood U of u, a unique vector of functions G such that $v = G(u)$ and $F(w, G(w)) = 0$ for all $w \in U$. In words, Eq. (13) uniquely determines Y as a function of X, locally around u. Solving for Y, given F and a value u for X, requires forming $\nabla_Y F$ as well as inverting it.

Structural BTF decomposition: One could instead avoid forming $\nabla_Y F$ by focusing on its *structural* nonsingularity, which only exploits the incidence graph $\mathcal{G}_F$ of system F ($\mathcal{G}_F$ is the bipartite graph having $F \uplus Y$ as set of vertices and an edge (f, y) if and only if variable y occurs in function f). A square matrix is said to be *structurally nonsingular* if it remains almost everywhere[3] nonsingular when its nonzero coefficients vary over some neighborhood. It has been

[3] Outside a set of values of Lebesgue measure zero.

shown (see for instance [14, 12, 16, 17]) that the Jacobian $\nabla_Y F$ is structurally nonsingular if and only if there exists a bijective assignment $\psi : Y \mapsto F$ such that $(\psi(y), y)$ is an edge of $\mathcal{G}_F$ for every $y \in Y$. Having this bijection we turn $\mathcal{G}_F$ into a directed graph $\vec{\mathcal{G}}_F$ by fixing the orientation $z \to \psi(y) \to y$ for every $z \neq y$ such that $(\psi(y), z) \in \mathcal{G}_F$. The strongly connected components of $\vec{\mathcal{G}}_F$ are called the *blocks* of F and are independent from the particular choice for ψ. Blocks are partially ordered by the order induced by $\vec{\mathcal{G}}_F$. The set of blocks of F equipped with this partial order is called the (structural) *Block Triangular Form* (BTF) decomposition of F [8].

Index reduction: For DAE, determining the leading variables as functions of the state variables (assuming a consistent initial value) requires finding all the latent equations, until the augmented system becomes a *semi-explicit* DAE:

$$\begin{cases} X' = G(X,Y) \\ 0 \;\; = F(X,Y) \end{cases} \quad \text{with } \nabla_Y F \text{ nonsingular,} \qquad (14)$$

so that the Implicit Function Theorem applies to F. The number of successive differentiations needed for getting this form is called the *differentiation index* [6] and the whole process is referred to as *index reduction*. Unlike ODEs, however, where the derivatives are explicitly given as functions of the state variables, simulating a semi-explicit DAE requires computing the Jacobian $\nabla_Y F$ and inverting it. Such computation will be performed eventually several times while searching for latent equations.

In practice, such brute force approach is ineffective and does not scale up. Tools handling DAE systems perform instead a *structural index* reduction, by exploiting the structural BTF decomposition of the involved Jacobians using the incidence graph of the system. The resulting procedure is called the *structural analysis* of DAE systems [14, 12, 17]. It may miss some numerical corner cases, but is computationally much more attractive than the full numerical approach. In the coming sections we extend the structural analysis to multi-mode systems, by handling continuous modes and events with their resets as equal citizens.

3.2 Multi-Mode DAE Systems

We now formally define the class of systems of multi-mode Differential/difference Algebraic Equations we are concerned with in this paper.

Consider a finite set of variables X; for $x \in X$ and $m \in \mathbb{N}$, the m-differentiation and m-shift of x are denoted by $x^{(\prime m)}$ and $x^{(\bullet m)}$, respectively. Let $X^{(\prime m)}$ and $X^{(\bullet m)}$ denote the set of all $x^{(\prime m)}$ and $x^{(\bullet m)}$, for x ranging over the set X of variables. We define:

$$X^{(\prime)} =_{\text{def}} \bigcup_{m \in \mathbb{N}} X^{(\prime m)} \text{ and } X^{(\bullet)} =_{\text{def}} \bigcup_{m \in \mathbb{N}} X^{(\bullet m)} \qquad (15)$$

Definition 2 *A* mDAE (multi-mode DAE system)*, resp.* mdAE (multi-mode dAE system)*, s is a tuple of n guarded equations:*

$$\begin{array}{rcl} s & =_{\text{def}} & e_1, \ldots e_n \\ e_i & =_{\text{def}} & \texttt{if } \gamma_i \texttt{ do } f_i = 0 \end{array}$$

where: X is a finite set of variables; f_i is a smooth scalar function over $X^{(\prime)}$, resp. $X^{(\bullet)}$; γ_i is a predicate over $X^{(\prime)}$, resp. $X^{(\bullet)}$.

In a mDAE or mdAE, a *mode* is a valuation in $\{\text{F}, \text{T}\}$ of its guards γ_i. In the guarded equation $(e_i) := (\texttt{if } \gamma_i \texttt{ do } f_i = 0)$, the equation $f_i = 0$ is enabled if and only if the guard γ_i holds. Otherwise the equation is disabled. Thus, a mode enables a subset of the equations $f_i = 0$ and disables the others.

A mDAE s_1 is transformed to a (nonstandard) mdAE s_2 through the following syntactic transformation:

$$s_2 \quad =_{\text{def}} \quad s_1 \left[x' \text{ is replaced by } \tfrac{x^\bullet - x}{\partial} \right] \qquad (16)$$

3.3 Structural Analysis of Multi-Mode Systems

The notion of constructive semantics was first introduced in the context of reactive synchronous programming languages [5, 3, 4], where it played an important role in grounding compilation on solid mathematical foundations. Essentially, a constructive semantics for a discrete time dynamical system consists of:

1. A specification of the set of *atomic actions*, which are effective, non-interruptible, state transformation operations. Executing an atomic action is often referred to as performing a *micro-step*;

2. A specification of the correct scheduling of the set of micro-steps constituting a *reaction*, by which discrete time progresses, from the current instant to the next one.

The principle of a constructive semantics is to decompose a time step into a sequence of micro-steps. The effect of atomic actions is to propagate the knowledge (value and status) regarding the state variables. For synchronous languages, atomic actions are restricted to either (i) the evaluation of a single expression, or (ii) control flow operations.

For mdAE systems, atomic actions comprise: (i) the evaluation of a guard; (ii) solving a block of numerical equations; (iii) equation management operations, for instance, adding a latent equation.

Observe that solving systems of mixed logico-numerical equations, involving a combination of guards and numerical variables, is not considered as an atomic action. The constructive semantics presented in this Section, requires that the evaluation of a guard γ_i precedes the resolution of the equation body $f_i = 0$.

3.3.1 Abstract Domain

The structural analysis method is based on an abstract semantics, in which numerical values are ignored and no numerical computation actually takes place. Instead, the abstract semantics defines a computation as an evolving knowledge regarding the statuses of the guards, variables and equations of an mdAE, namely:

- A guard may be *not evaluated*, *evaluated to true* or *evaluated to false*;
- A variable may be *undefined*, or *defined*;
- An equation may be *not evaluated*, *disabled*, or *evaluated*.

Unlike mono-mode DAE, the set of equations describing the current status of an mDAE is mode dependent and evolves therefore dynamically. To capture this important fact, we tag as irrelevant all those equations that are not currently involved. Formally, the semantics defines computations in a partially ordered finite domain of values D:

$$D = \{\mathrm{I}, \mathrm{U}, \mathrm{F}, \mathrm{T}\} \quad \text{with} \quad \mathrm{I} < \mathrm{U} < \mathrm{F}, \mathrm{T} \tag{17}$$

The meaning of these values is as follows:

- The minimal element I is used to represent the fact that a variable, a guard, or an equation is *irrelevant*, that is not used to define the current status of the mdAE system.
- Value U means that a variable, guard or equation has not been evaluated yet (say it is *undefined*). 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. This value does not apply to variables.

The constructive semantics defines the allowed micro-steps as a non-deterministic transition relation between abstract states, called *statuses*.

Definition 3 (Status) *The set V of* S-variables *is defined by*

$$\begin{aligned} V \quad =_{\text{def}} \quad & \left\{x^{(\bullet m)}\right\}_{x\in X, m\in\mathbb{N}} \\ \cup \quad & \left\{\gamma_i\right\}_{i=1\ldots n} \\ \cup \quad & \left\{e_i^{(\bullet m)}\right\}_{i=1\ldots n, m\in\mathbb{N}} \end{aligned}$$

A status σ *is a valuation in* D *of the S-variables, that is a mapping* $V \to D$*. A status* $\sigma : V \to D$ *is said to be* finite *if it is almost everywhere equal to* I*. The set of statuses is partially ordered by the product order:* $\sigma_1 \leq \sigma_2$ *if and only if for all* $v \in V$*,* $\sigma_1(v) \leq \sigma_2(v)$*.*

The partial order relation on statuses plays an important role to guarantee that knowledge increases at every micro-step of the semantics. This is ensured by the fact that the transition relation is strictly monotonous.

Coherence conditions: We define the *incidence graph* $\rho \subseteq V \times V$ of a mdAE system s as follows:

$$\begin{aligned} \left(\gamma_i, x^{(\bullet m)}\right) \in \rho \quad &\text{iff} \quad x^{(\bullet m)} \text{ appears in } \gamma_i \\ \left(e_i^{(\bullet p)}, x^{(\bullet m)}\right) \in \rho \quad &\text{iff} \quad x^{(\bullet m)} \text{ appears in } f_i^{(\bullet p)} \end{aligned}$$

Given a guard γ_i, $\rho(\gamma_i)$ is the set of variables $x^{(\bullet m)}$ appearing in γ_i. Given equation $e_i^{(\bullet p)}$, $\rho(e_i^{(\bullet p)})$ is the set of variables $x^{(\bullet m)}$ appearing in $f_i^{(\bullet p)}$.

The constructive semantics follows a causality principle, namely that an equation can not be solved before its guard has been evaluated true. Similarly, a guard can not be evaluated before all its incident variables have been defined. This results in the following *coherence* property which is an invariant of the constructive semantics: A status σ is *coherent* if and only if the following properties hold:

$$\begin{aligned} \left(\gamma_i, x^{(\bullet m)}\right) \in \rho \text{ and } \sigma\left(x^{(\bullet m)}\right) \leq \text{U} &\Rightarrow \sigma(\gamma_i) \leq \text{U} \\ \left(e_i^{(\bullet p)}, x^{(\bullet m)}\right) \in \rho \text{ and } \sigma\left(x^{(\bullet m)}\right) \leq \text{U} &\Rightarrow \sigma\left(e_i^{(\bullet p)}\right) \leq \text{F} \\ \sigma(\gamma_i) \leq \text{U} &\Rightarrow \sigma\left(e_i^{(\bullet m)}\right) \leq \text{U} \end{aligned}$$

Enabled Sets, Shifting Degree, Leading Variables: Given a coherent status σ, $i = 1 \ldots n$, guard γ_i is *enabled in* σ if and only if for all $x^{(\bullet m)} \in \rho(\gamma_i)$, $\sigma(x^{(\bullet m)}) = \text{T}$. Given a coherent status σ, $i = 1 \ldots n$ and $m \in \mathbb{N}$, equation $e_i^{(\bullet m)}$ is *enabled in* σ (respectively *disabled in* σ) if and only if $\sigma(\gamma_i) = \text{T}$ (respectively $\sigma(\gamma_i) = \text{F}$), where γ_i is the guard of $e_i^{(\bullet m)}$. Denote by $En_\gamma(\sigma)$ the set of guards that are enabled in σ, and by $En_f(\sigma)$ (respectively $Dis_f(\sigma)$) the set of equations that are enabled (respectively disabled) in σ. Notice that for any finite status σ, these sets are finite.

Denote by $Undef(\sigma) =_{\text{def}} \{v \in V \mid \sigma(v) \leq \text{U}\}$ the set of S-variables that are either irrelevant or undefined in status σ.

Define $d_\sigma^o(x)$, the *shifting degree* of x in σ, to be the least upper bound of the shifting degree m of all variables $x^{(\bullet m)}$ that are incident to an equation enabled in σ:

$$d_\sigma^o(x) =_{\text{def}} \sup \left\{ m \;\middle|\; \begin{array}{l} \exists i = 1 \ldots n, p \in \mathbb{N} \text{ s.t.} \\ e_i^{(\bullet p)} \in En_\gamma(\sigma) \text{ and} \\ x^{(\bullet m)} \in \rho\left(e_i^{(\bullet p)}\right) \end{array} \right\}$$

Notice that the shifting degree $d_\sigma^o(x) = -\infty$ if x is not incident to any enabled equation in σ. The shifting degrees in a finite status are bounded: given a finite status σ, there exists $N \in \mathbb{N}$ such that $d_\sigma^o(x) \leq N$ for all $x \in X$.

Given a status σ, the set of *leading variables* in status σ is the set of variables of maximal shifting degree that are incident to an enabled equation:

$$Ld(\sigma) =_{\text{def}} \left\{ x^{(\bullet m)} \;\middle|\; x \in X \text{ and } m = d_\sigma^o(x) \geq 0 \right\}$$

Contexts: The constructive semantics must also take into account possible *consistency equations*. This is the purpose of *contexts*, exemplified in Exec. Sch. 4 (Section 2). A *context*

$$\Delta \subseteq \left\{ e_i^{(\bullet m)} \right\}_{i=1\ldots n, m \in \mathbb{N}}$$

is a set of equations involving no leading variable in the considered status. Given a context Δ, equation $e_i^{(\bullet m)} \in \Delta$ is assumed to be satisfied, as soon as its guard γ_i has been evaluated to true. In this case, the constructive semantics sets such an equation as being solved, without actually solving the equation. That this equation is satisfied is known from the previous time step.

3.3.2 Constructive Semantics

Given a finite coherent initial status σ_0, and a finite context Δ, the *constructive semantics* of a mdAE system s is the set of the finite increasing sequences of statuses, called *runs*:

$$\sigma_0 < \sigma_1 < \cdots < \sigma_k < \sigma_{k+1} < \cdots < \sigma_K \qquad (18)$$

such that for every $k < K$, the pair (σ_k, σ_{k+1}) is a micro-step in the context Δ. A micro-step transforms status σ_k into status σ_{k+1} by updating the values of a bounded subset of S-variables, from U to T or F, or from I to U, via some atomic action.

Definition 4 *A run $\sigma_0 < \ldots < \sigma_K$ is called* successful *if and only if in status σ_K is* successful*, that is all equations e_i have either the value* T *or* F *and no leading variable has the value* U*. The constructive semantics* succeeds *for an initial status σ_0 and context Δ if it has, for every mode, at least one successful run.*

When a run is successful, the system can proceed to the next time step, by executing a *Tick* micro-step, where, in a nutshell, time is advanced and defined variables are shifted. Algorithm 5 defines the computation of a micro-step from a given status σ and context Δ. To produce a run, Algorithm 5 should be iterated, until a *Tick* micro-step is performed.

The algorithm starts with a finite coherent status σ and a context Δ. The context Δ is the (possibly empty) set of equations known to be satisfied by the defined values in the current time-step. Notice that the context is updated at each *Tick*.

Line 1: Function $Success(\sigma)$ decides whether status σ is successful, according to Definition 4.

Line 2: If the status is deemed successful, a *Tick* micro-step is performed. This has the effect of shifting backward defined variables, and setting all other S-variables $v \in V$, either to U, if v is in the mdAE s, or I, otherwise. The new context is defined to be the set of equations that are known to be satisfied. Formally

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

where:

$$\begin{array}{rcl} \sigma^\circ(\gamma_i) & = & \text{U} \\ \sigma^\circ\big(x^{(\bullet m)}\big) & = & \textbf{if } \sigma\big(x^{(\bullet m+1)}\big) = \text{T} \textbf{ then T} \\ & & \textbf{else if } x^{(\bullet m)} \text{ is a variable of mdAE } s \\ & & \textbf{then U else I} \\ \sigma^\circ\big(e^{(\bullet m)}\big) & = & \textbf{if } e^{(\bullet m)} \text{ is a variable of } s \textbf{ then U else I} \end{array}$$

and

$$\Delta^\circ = \left\{ e_i^{(\bullet m)} \;\middle|\; \begin{array}{l} \exists j = 1 \ldots n, f_j \text{ is} \\ \text{syntactically identical to } f_i \\ \text{and } \sigma\big(e_j^{(\bullet m+1)}\big) = \text{T} \end{array} \right\}$$

Algorithm 5 Computation of a time step; Atomic Actions are written in sans-serif font.

Require: a finite coherent status σ, and a finite context Δ; **return** (updated) σ and Δ

```
1:  if Success(σ) then
2:      (σ, Δ) ← Tick(σ)
3:  else
4:      F ← En_f(σ) ∩ Undef(σ)
5:      if exists B ∈ Blocks(F) then
6:          σ ← EvaluateBlock(B, σ)
7:      else
8:          if exists γ_i ∈ En_γ(σ) ∩ Undef(σ) then
9:              σ ← EvaluateGuard(γ_i, σ)
10:             σ ← DisableEquation(γ_i, σ)
11:             σ ← EvaluateRedundant(γ_i, Δ, σ)
12:         else
13:             if exists e_i^(•m) ∈ Overdetermined(F)
    then
14:                 σ ← ForwardShift(e_i^(•m), σ)
15:             else
16:                 L ← LatentEquations(F)
17:                 if L = ∅ then
18:                     Fail(σ)
19:                 else
20:                     σ ← AddEquation(L, σ)
```

Note that *Tick* is not increasing (actually it does not have to be so, since it applies when moving to the next time step).

Line 4: The system F collects the enabled guarded equations in the status σ that are still undefined. By applying the procedure BTF (Section 3.1) to F one gets three distinct sets: $B_{\mathtt{ns}}$, $B_{\mathtt{o}}$, and $B_{\mathtt{u}}$, the enabled, overdetermined, and underdetermined blocks, respectively. We further apply a post processing step to the standard BTF: for the overdetermined subsystem, we select a maximum square triangular submatrix and append it to $B_{\mathtt{ns}}$ to obtain $Blocks(F)$ (Line 5). Function *Overdetermined* (Line 13) returns what is left in $B_{\mathtt{o}}$. For instance, for the system $F := \{f_1(x_1){=}0, f_2(x_1){=}0\}$, BTF gives $B_{\mathtt{u}} = B_{\mathtt{ns}} = \emptyset$ and $B_{\mathtt{o}} = \{f_1{=}0, f_2{=}0\}$. We match arbitrarily either f_1 or f_2 to x_1. We therefore get $Blocks(F) = \{f_1{=}0\}$, and $Overdetermined(F) = \{f_2{=}0\}$. The impact of

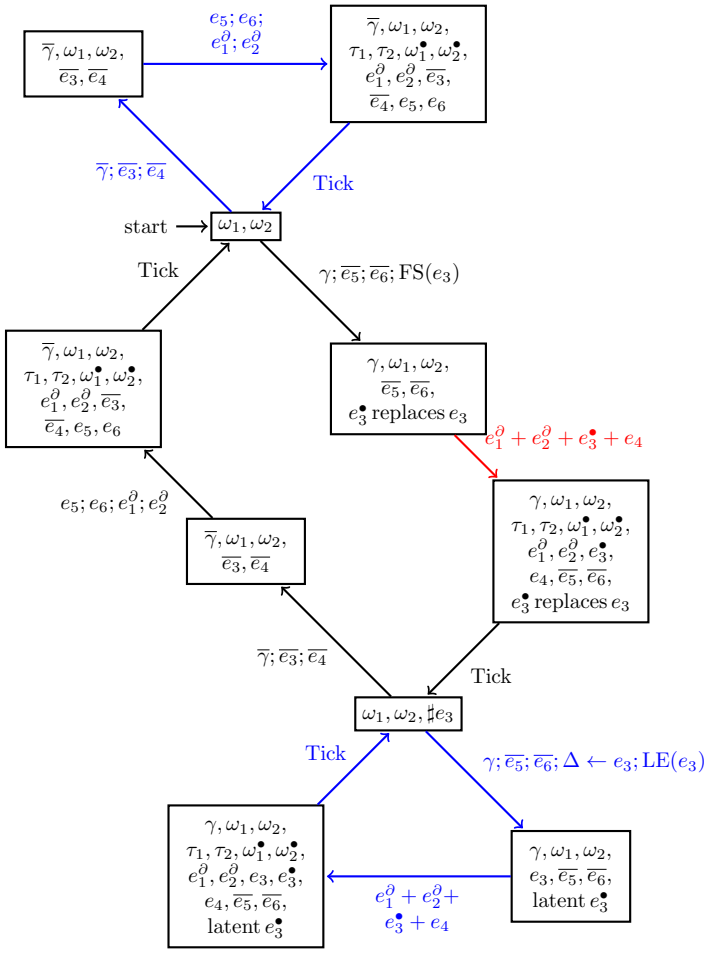


Figure 3: Constructive semantics of the Simple Clutch. Notations: For all statuses (shown in boxes), v (resp. $\overline{v}$) means $v = \mathrm{T}$ (resp. $v = \mathrm{F}$), and not mentioning v means $v = \mathrm{U}$. $\sharp e$ means that e_f belongs to context Δ. FS(.) (resp. LE(.); resp. $\Delta \leftarrow .$) refers to line 14, forward shift (resp. 16, latent equation; resp. 11, redundent equations) of Algorithm 5. Blue (resp. black) transitions belong to a continuous-time (resp. discrete-time) dynamics. The red transition is impulsive. A semicolon is the sequential composition of micro-steps, and the $+$ sign denotes blocks of equations.

the different possible choices on the simulation of the system is left as a future work.

Line 6: The atomic action EvaluateBlock(B, σ) solves block B for its dependent variables, hence, it updates the status σ to reflect that the undefined variables and equations involved in B become defined. Formally, in the resulting new status σ',

$$\begin{aligned} \forall e_i^{(\bullet p)} \in B \quad &\Longrightarrow \quad \sigma'\big(e_i^{(\bullet p)}\big) = \mathrm{T} \\ \forall v \in \rho\big(e_i^{(\bullet p)}\big) \quad &\Longrightarrow \quad \sigma'(v) = \mathrm{T} \end{aligned}$$

Line 8: Select one enabled but undefined guard γ_i, and evaluates it to T or F (*Line 9*). Both alternatives must be explored, and an implementation will fork

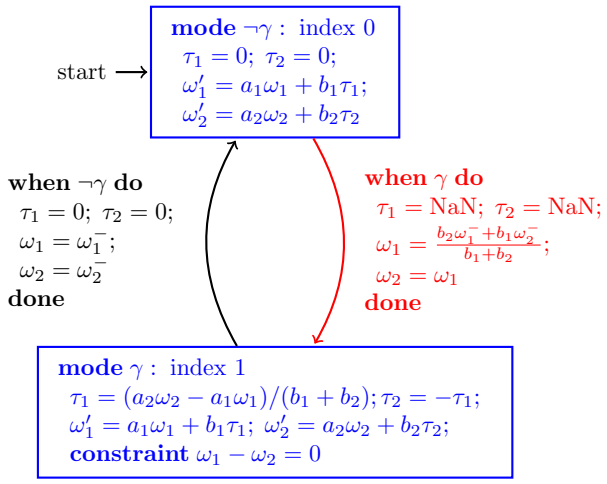


Figure 4: Standardization of the clutch's constructive semantics. Blocks have been standardized and then symbolically pivoted. x^- is the previous value of state variable x, which is the left limit of x when exiting a mode. Continuous-time dynamics are colored blue; non-impulsive (resp. impulsive) state-jumps 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).

two child Micro-Step procedures to explore the graph of all possible runs. Such implementation details are out of scope for this paper.

Line 10: If guard γ_i is evaluated to F, the equations it controls is disabled (set to F).

Line 11: The context Δ is used to update the status σ through the atomic action EvaluateRedundant. For the freshly evaluated guard γ_i, all its corresponding equations $e_i^{(\bullet m)}$ belonging to the context Δ are set to evaluated (value T). Equations $e_i^{(\bullet m)} \notin \Delta$ remain unchanged.

Line 14: The atomic action ForwardShift attempts to relax an overdetermined system F by shifting one blocking (overdetermined) equation at a time.

Definition 5 (Forward Shift) *The* forward shift *of equation* $e_i^{(\bullet m)} =_{\text{def}} \texttt{if } \gamma_i \texttt{ do } f_i^{(\bullet m)} = 0$, *is defined by*

$$e_i^{(\bullet m+1)} =_{\text{def}} \texttt{if } \gamma_i \texttt{ do } f_i^{(\bullet m+1)} = 0$$

where $f_i^{(\bullet k)}$ *amounts to shifting forward* k*-times the arguments of* f_i. *Notice that only the body of the equation is shifted, not its guard.*

Forward shifting equation $e_i^{(\bullet m)}$ updates the status from σ to σ' as follows:

$$\sigma\big(e_i^{(\bullet m)}\big) = \text{U} \quad \text{becomes} \quad \sigma'\big(e_i^{(\bullet m)}\big) = \text{F}$$
$$\sigma\big(e_i^{(\bullet m+1)}\big) = \text{I} \quad \text{becomes} \quad \sigma'\big(e_i^{(\bullet m+1)}\big) = \text{U}$$

which is increasing.

Line 16: Exhibiting latent equations is a classical task since we are just dealing with a dAE (difference Algebraic Equation) system. We can, e.g., use the Pantelides algorithm [14] or the Σ-method of [17], which also identifies when the index is infinite. Indeed, the algorithm rejects models with infinite structural index (*Lines 17 and 18*). Intuitively, this problem occurs when exhibiting latent equations results in introducing at least as many extra variables as new equations making the perfect matching problem unsolvable in finitely many steps.

Line 20: The atomic action AddEquation augments the considered underdetermined block by adding the latent equations, i.e., it extends the support of the status σ with the finitely many extra latent equations in L such that the newly obtained status is coherent and $\sigma(v) > 1$ for all $v \in L$.

Properties of the Constructive Semantics: Algorithm 5 is iterated in order to generate all possible runs, corresponding to the different modes of the system. This is done until all reachable pairs (σ, Δ) of statuses and contexts have been explored.

As a result, we obtain the Constructive Semantics in the form of a graph **CS** having as nodes the different encountered status-context pairs and as edges the micro-steps. Elementary cycles of **CS** 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, we call them *reset runs.* Elementary cycles of **CS** containing at least two reset runs and having stationary assignments of the guards correspond to an execution looping forever, in an attempt to handle a mode change: a model exhibiting this situation is rejected—see Appendix B for a simple example.

In Figure 3, we depict the graph **CS** produced for the clutch example and Figure 4 shows the effective code resulting from the standardization of **CS**.

4 Conclusion

We propose a formal approach for the structural analysis of multi-mode DAE systems that extends and adapts the dummy derivatives method of [12]. We further complement our analysis with a standardization step leading, when successful, to execution schemes that could be used for numerical simulations. The use of nonstandard analysis was essential in defining an operational semantics when discrete events occur. We see our work as a generalization of adequate formalizations where only ODEs are involved [9].

We identified several interesting avenues for future work. In particular, we plan to work on generic standardization techniques to handle a larger class of problems. This is a crucial step for our structural analysis to be useful in practice. The exact computation of standard finite solutions has the advantage of giving exact reset maps at events of mode changes. It, however, requires symbolic manipulations and could therefore be computationally expensive. A viable and relatively cheaper approach would be to use numerical approximations where the infinitesimals are substituted by small real numbers. In this case, one has to rely on sufficient conditions to prove the existence of the standard solutions and to study further the accuracy and the effect of their numerical approximations on subsequent computations.

We are also currently implementing Algorithm 5 to assess its performance on real case studies. The prototype will help us studying the confluence of local nondeterministic choices when handling overdetermined modes and, more importantly, their effect on the overall simulation.

References

[1] A. Benveniste, T. Bourke, B. Caillaud, and M. Pouzet. Nonstandard semantics of hybrid systems modelers. *J. Comput. Syst. Sci.*, 78(3):877–910, 2012.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] S. L. Campbell and C. W. Gear. The index of general nonlinear DAEs. *Numer. Math.*, 72:173–196, 1995.

[7] J. Dieudonné. *Fondements de l'analyse moderne*. Gauthier-Villars, 1965.

[8] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices*. Numerical Mathematics and Scientific Computation. Oxford University Press, 1986.

[9] E. A. Lee. Constructive models of discrete and continuous physical phenomena. *IEEE Access*, 2:797–821, 2014.

[10] T. Lindstrøm. An invitation to nonstandard analysis. In N. Cutland, editor, *Nonstandard Analysis and its Applications*, pages 1–105. Cambridge Univ. Press, 1988.

[11] 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.

[12] S. E. Mattsson and G. Söderlind. Index reduction in Differential-Algebraic Equations using dummy derivatives. *Siam J. Sci. Comput.*, 14(3):677–692, 1993.

[13] 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.

[14] C. Pantelides. The consistent initialization of differential-algebraic systems. *SIAM J. Sci. Stat. Comput.*, 9(2):213–231, 1988.

[15] 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.

[16] A. Pothen and C. Fan. Computing the block triangular form of a sparse matrix. *ACM Trans. Math. Softw.*, 16(4):303–324, 1990.

[17] J. D. Pryce. A simple structural analysis method for DAEs. *BIT*, 41(2):364–394, 2001.

A Standardization

We mechanize below the manual reasoning performed in Section 2 for a larger class of continuous functions.

A.1 Impulse Analysis

The impulse analysis consists in abstracting hyperreals with their magnitude order (or simply "order") compared to the infinitesimal ∂. The order of the hyperreal x, denoted by $[x]$, is defined as the integer $n \in \mathbb{Z}$, if it exists, such that the standard part of $x.\partial^n$ is a nonzero finite real number. By convention, the order of 0 is $-\infty$.

For instance, the order of any nonzero real number, seen as a hyperreal, is 0. Multiplying x by ∂^m, for some integer m shifts $[x]$ by $-m$: $[x.\partial^m] := -m + [x]$. The order for a monomial function is given by $[x_1^{r_1} \cdots x_n^{r_n}] = \sum_{i=1}^{n} r_i[x_i]$. For a multivariate polynomial function, the order is the maximum of the orders of all its monomials with highest total degree, and, for a rational function $\frac{P}{Q}$, the order is $[P] - [Q]$. For instance, the order of a linear function $f(x_1, \ldots, x_n)$ is

$$[f(x_1, \ldots, x_n)] = \max_{i \in [1,\ldots,n]} [x_i] \ . \tag{19}$$

whereas the order of $f(x_1, x_2) := x_1 + x_1 x_2 + x_2^2$ is $\max\{[x_1] + [x_2], 2[x_2]\}$. We leave the general case for continuous functions as a future work.

We develop below the impulse analysis for the two transitions $\gamma : \text{T} \to \text{F}$ and $\gamma : \text{F} \to \text{T}$ of System (9) assuming linear f_i as in Eq. (11).

Mode change $\gamma : \text{T} \to \text{F}$*:* Recall that when γ goes from T to F, we obtain a system of 4 equations $(e_1^\partial, e_2^\partial, e_5, e_6)$ for 4 unknowns $(\tau_1, \tau_2, \omega_1^\bullet, \omega_2^\bullet)$ and we assume that the state variables ω_1 and ω_2 are known and finite. Thus, $[\omega_i] \leq 0$ (we use an inequality to take into account the special case $\omega_i = 0$, in which case the order would be $-\infty$). This yields the following abstraction ($i = 1, 2$):

$$\left\{ \begin{array}{rcll} [\omega_i^\bullet - \omega_i] & = & -1 + [f_i] & ([e_i^\partial]) \\ [\tau_1] & = & -\infty & ([e_5]) \\ [\tau_2] & = & -\infty & ([e_6]) \end{array} \right. \tag{20}$$

In (20), since f_i, $i = 1, 2$, are linear, $[f_i] = \max\{[\omega_i], [\tau_i]\}$ (cf. Eq. (19)), and therefore, $[f_i] \leq [\omega_i] \leq 0$. We are interested in the order of the difference $\omega_i^\bullet - \omega_i$, regarded as a single hyperreal. Eq. (20) thus gives $[\omega_i^\bullet - \omega_i] = -1 + [f_i] \leq -1 + [\omega_i] \leq -1$ and we conclude that the transition is continuous in ω_i.

Mode change $\gamma : \text{F} \to \text{T}$*:* Similar to the previous case, we also assume that the values of ω_i are known and are finite from the previous step. Thus $[\omega_i] \leq 0$. When γ becomes T, the new state may not satisfy $\omega_1 - \omega_2 = 0$, since $(eq_3^\bullet)$ was not active in previous mode ($\gamma = \text{F}$). We eliminate, in the system of Line 3 in Exec. Sch. 4, $(eq_3^\bullet)$ and (eq_4) by setting $\omega^\bullet =_{\text{def}} \omega_1^\bullet = \omega_2^\bullet$ and $\tau =_{\text{def}} \tau_1 = -\tau_2$, which yields

$$\left\{ \begin{array}{ll} \omega^\bullet - \omega_1 = \partial.f_1(\omega_1, \tau) & (eq_1^\partial) \\ \omega^\bullet - \omega_2 = \partial.f_2(\omega_2, \tau) & (eq_2^\partial) \end{array} \right. \tag{21}$$

Using (19), the impulse analysis for the simplified system yields, for $i = 1, 2$:

$$[\omega^\bullet - \omega_i] = -1 + \max\{[\omega_i], [\tau]\}$$

At this point, two cases can occur: if $[\tau] \leq 0$, then $[\omega^\bullet - \omega_i] \leq -1$ for $i = 1, 2$, which is not possible since it would require $\omega_1 = \omega_2$, which does not hold in general. Thus, $[\tau] \geq 1$ and τ is impulsive. This implies $[\omega^\bullet - \omega_i] \geq 0$, expressing impulsive torques and discontinuous angular velocities.

A.2 Computation of Resets

In this section we mechanize the computation of the resets. We replace the manual rewriting used in Section 2.4 by a calculus on formal power series. In (21), we now regard the leading variables $\omega^\bullet, \tau$, as formal power series in the variable ∂^{-1}. The support of these series is determined by the impulse analysis developed in Appendix A.1:

$$\begin{array}{rcl} \omega^\bullet & = & \displaystyle\sum_{k=0}^{\infty} \omega_k^\bullet \, \partial^k \\ \tau & = & \partial^{-1} \displaystyle\sum_{k=0}^{\infty} \tau_k \, \partial^k \end{array} \tag{22}$$

where all coefficients $\omega_k^\bullet, \tau_k$ are finite. Using this expansion and the linearity of the f_i, (21) becomes

$$\sum_{k=0}^{\infty} \omega_k^\bullet \, \partial^k - \omega_1 = \partial. \left(a_1\omega_1 + b_1 \left(\partial^{-1} \sum_{k=0}^{\infty} \tau_k \, \partial^k \right) \right)$$
$$\sum_{k=0}^{\infty} \omega_k^\bullet \, \partial^k - \omega_2 = \partial. \left(a_2\omega_2 - b_2 \left(\partial^{-1} \sum_{k=0}^{\infty} \tau_k \, \partial^k \right) \right)$$

We standardize this system by keeping only the dominant terms:

$$\left\{ \begin{array}{rcl} \omega_0^\bullet - \omega_1 & = & b_1\tau_0 \\ \omega_0^\bullet - \omega_2 & = & -b_2\tau_0 \end{array} \right. \tag{23}$$

It remains to solve this system for the standard variables (coefficients) $\omega_0^\bullet, \tau_0$). Thus,

$$\omega_0^\bullet = \frac{b_2\omega_1 + b_1\omega_2}{b_1 + b_2} \tag{24}$$

and our analysis is complete.

Dividing the value τ_0 for the solution of (23) by the actual (non infinitesimal) step size δ used, yields an estimate of the Dirac impulse for the torque, integrated over the time interval of length δ. It would be interesting to study the accuracy of this estimate.

B Overdetermined Example

We show in this appendix how Algorithm 5 behaves on a simple overdetermined example, where only static equations occur:

$$S \;:\; \left\{ \begin{array}{rcll} f_1(x_1, x_2) & = & 0 & (e_1) \\ f_2(x_1, x_2) & = & 0 & (e_2) \\ f_3(x_1, x_2) & = & 0 & (e_3) \end{array} \right.$$

Equations are not guarded, equivalently, we regard all the guards as being true. Thus, the three equations are active and x_1, x_2 are the leading variables. The following comments refer to lines of Algorithm 5.

Initialization: At the initialization of the step, nothing is evaluated and the context is empty:

$$\begin{array}{rcl} \sigma(x_1, x_2) & = & (\mathrm{U}, \mathrm{U}) \\ \sigma(e_1, e_2, e_3) & = & (\mathrm{U}, \mathrm{U}, \mathrm{U}) \\ \Delta & = & \emptyset \end{array} \tag{25}$$

and shifted versions of the above variables and equations have all the value I (irrelevant). Since the resulting status is not successful, we move to Line 4.

Line 4: F is the whole system S. We thus apply BTF on the whole system S, which returns $B_{\mathtt{u}} = B_{\mathtt{ns}} = \emptyset$ and $B_{\mathtt{o}} = S$, expressing that the entire system is overdetermined. Performing the additional processing of Line 4 returns, say, an enabled block $\{e_1, e_2\}$ with dependent variables x_1, x_2 and an overdetermined equation (e_3) having no dependent variables. We thus go to Line 5, which brings us to Line 6.

Line 6: We solve block $\{e_1, e_2\}$ for x_1, x_2 and move to the next micro-step with status and context

$$\begin{array}{rcl} \sigma(e_1, e_2) & = & (\mathrm{T}, \mathrm{T}) \; ; \; \sigma(e_3) \; = \; \mathrm{U} \\ \sigma(x_1, x_2) & = & (\mathrm{T}, \mathrm{T}) \\ \Delta & = & \emptyset \end{array}$$

and other shifted S-variables being irrelevant.

Line 4: F is the singleton system $\{e_3\}$ with no dependent variable. We thus go to Line 13, which brings us to Line 14.

Line 14: We apply ForwardShift, which amounts to replacing e_3 by $e_3^\bullet$, having $x_1^\bullet, x_2^\bullet$ as dependent variables, and we move to the next micro-step with status and context

$$\begin{array}{rcl} \sigma(e_1, e_2) & = & (\mathrm{T}, \mathrm{T}) \; ; \; \sigma(e_3) \; = \; \mathrm{F} \; ; \; \sigma(e_3^\bullet) \; = \; \mathrm{U} \\ \sigma(x_1, x_2) & = & (\mathrm{T}, \mathrm{T}) \; ; \; \sigma(x_1^\bullet, x_2^\bullet) \; = \; (\mathrm{U}, \mathrm{U}) \\ \Delta & = & \emptyset \end{array}$$

and other shifted S-variables being irrelevant.

Line 4: F is the system $\{e_3^\bullet\}$ with $x_1^\bullet, x_2^\bullet$ as dependent variables. $\{e_3^\bullet\}$ is underdetermined, so we go to Line 16.

Line 16: Find latent equations in the system $\{e_1, e_2, e_3^\bullet\}$ having dependent variables $x_1, x_2, x_1^\bullet, x_2^\bullet$. Take $e_1^\bullet$ as latent equation and add it to the system. Move to the next micro-step with status and context

$$\begin{array}{rcl} \sigma(e_1, e_2) & = & (\mathrm{T}, \mathrm{T}) \; ; \; \sigma(e_3) = \mathrm{F} \; ; \; \sigma(e_1^\bullet, e_3^\bullet) = (\mathrm{U}, \mathrm{U}) \\ \sigma(x_1, x_2) & = & (\mathrm{T}, \mathrm{T}) \; ; \; \sigma(x_1^\bullet, x_2^\bullet) = (\mathrm{U}, \mathrm{U}) \\ \Delta & = & \emptyset \end{array}$$

and other shifted S-variables being irrelevant.

Line 4: F is the system $\{e_1^\bullet, e_3^\bullet\}$ with $x_1^\bullet, x_2^\bullet$ as dependent variables. BFT returns a single enabled block, which we evaluate. The micro-step ends with the *successful* status and context

$$\begin{array}{rcl} \sigma(e_1, e_2, e_1^\bullet, e_3^\bullet) & = & (\mathrm{T}, \mathrm{T}, \mathrm{T}, \mathrm{T}) \ ; \ \sigma(e_3) \ = \ \mathrm{F} \\ \sigma(x_1, x_2, x_1^\bullet, x_2^\bullet) & = & (\mathrm{T}, \mathrm{T}, \mathrm{T}, \mathrm{T}) \\ \Delta & = & \emptyset \end{array}$$

and other shifted S-variables being irrelevant.

Line 2: We thus perform a *Tick* by forming the initial status σ and context Δ for the next time step:

$$\begin{array}{rcll} \sigma(x_1, x_2) & = & (\mathrm{T}, \mathrm{T}) & \\ \sigma(e_1, e_2, e_3) & = & (\mathrm{U}, \mathrm{U}, \mathrm{U}) & (26) \\ \Delta & = & \{e_1, e_3\} & \end{array}$$

Observe that (26) differs from (25). So we are not in a continuous mode. Since Δ is not empty and guards are all true, we move to Line 11.

Line 11: Update

$$\sigma(e_1, e_3) \quad \leftarrow \quad (\mathrm{T}, \mathrm{T})$$

So we are left with (e_2) overdetermined, so we go to Line 13, which brings us to Line 14.

Line 14: We apply ForwardShift, which amounts to replacing e_2 by $e_2^\bullet$, having $x_1^\bullet, x_2^\bullet$ as dependent variables. Now, the leading variables become $x_1^\bullet, x_2^\bullet$.

Line 4: F is the system $\{e_2^\bullet\}$ with $x_1^\bullet, x_2^\bullet$ as dependent variables. $\{e_2^\bullet\}$ is underdetermined, so we go to Line 16.

Line 16: Find latent equations in the system $\{e_1, e_2^\bullet, e_3\}$ having dependent variables $x_1, x_2, x_1^\bullet, x_2^\bullet$. Take $e_1^\bullet$ as latent equation and add it to the system. Move to the next micro-step with status and context

$$\begin{array}{rcl} \sigma(e_1, e_3) & = & (\mathrm{T}, \mathrm{T}) \ ; \ \sigma(e_2) = \mathrm{F} \ ; \ \sigma(e_1^\bullet, e_2^\bullet) = (\mathrm{U}, \mathrm{U}) \\ \sigma(x_1, x_2) & = & (\mathrm{T}, \mathrm{T}) \ ; \ \sigma(x_1^\bullet, x_2^\bullet) = (\mathrm{U}, \mathrm{U}) \\ \Delta & = & \emptyset \end{array}$$

and other shifted S-variables being irrelevant. Go to Line 4.

Line 4: F is the system $\{e_1^\bullet, e_2^\bullet\}$ with $x_1^\bullet, x_2^\bullet$ as dependent variables. BFT returns a single enabled block, which we evaluate. The micro-step ends with the *successful* status and context

$$\begin{array}{rcl} \sigma(e_1, e_3, e_1^\bullet, e_2^\bullet) & = & (\mathrm{T}, \mathrm{T}, \mathrm{T}, \mathrm{T}) \ ; \ \sigma(e_2) \ = \ \mathrm{F} \\ \sigma(x_1, x_2, x_1^\bullet, x_2^\bullet) & = & (\mathrm{T}, \mathrm{T}, \mathrm{T}, \mathrm{T}) \\ \Delta & = & \emptyset \end{array}$$

and other shifted S-variables being irrelevant.

Line 2: We thus perform a *Tick* by forming the initial status σ and context Δ for the next time step:

$$\begin{array}{rcll} \sigma(x_1, x_2) & = & (\mathrm{T}, \mathrm{T}) & \\ \sigma(e_1, e_2, e_3) & = & (\mathrm{U}, \mathrm{U}, \mathrm{U}) & (27) \\ \Delta & = & \{e_1, e_2\} & \end{array}$$

Observe that (27) differs from (26). So we are not in a continuous mode. Since Δ is not empty and guards are all true, we move to Line 11.

This goes further on. In finitely many rounds we end up having a cycle, shown Figure 5. This cycle has stationary guards (constantly true). The model is rejected.

Of course, any DAE tool would immediately identify S as being overconstrained. Our algorithm looks like a significant overshoot for this case. Its merit, however, is that it applies to all models. Clearly, nothing forbids us to shorten the decisions by applying obvious heuristics to simple cases.

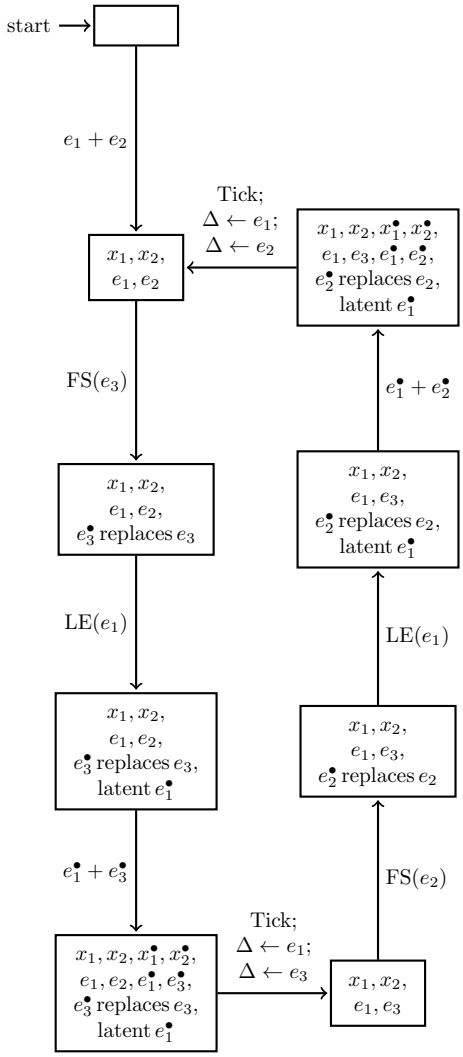
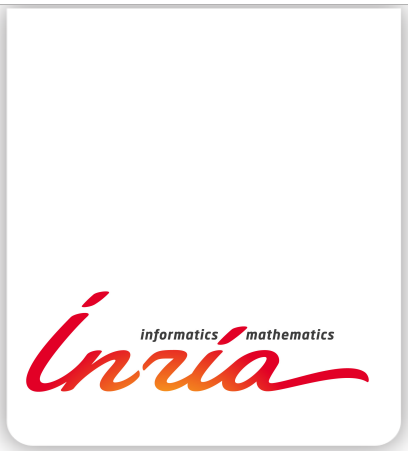


Figure 5: Constructive semantics of the overdetermined system. *Notations are those of Fig. 3.*



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