Logical Methods in Computer Science
Volume 15, Issue 1, 2019, pp. 14:1–14:27
https://lmcs.episciences.org/

Submitted Oct. 24, 2017
Published Feb. 15, 2019

ALGEBRA, COALGEBRA, AND MINIMIZATION IN POLYNOMIAL DIFFERENTIAL EQUATIONS

MICHELE BOREALE

Università di Firenze, Dipartimento di Statistica, Informatica, Applicazioni (DiSIA) "G. Parenti", Viale Morgagni 65, I-50134 Firenze, Italy.
e-mail address: michele.boreale@unifi.it

Abstract. We consider reasoning and minimization in systems of polynomial ordinary differential equations (ODE's). The ring of multivariate polynomials is employed as a syntax for denoting system behaviours. We endow this set with a transition system structure based on the concept of *Lie derivative*, thus inducing a notion of $\mathcal{L}$-*bisimulation*. We prove that two states (variables) are $\mathcal{L}$-bisimilar if and only if they correspond to the same solution in the ODE's system. We then characterize $\mathcal{L}$-bisimilarity algebraically, in terms of certain ideals in the polynomial ring that are invariant under Lie-derivation. This characterization allows us to develop a complete algorithm, based on building an ascending chain of ideals, for computing the largest $\mathcal{L}$-bisimulation containing all valid identities that are instances of a user-specified template. A specific largest $\mathcal{L}$-bisimulation can be used to build a reduced system of ODE's, equivalent to the original one, but *minimal* among all those obtainable by linear aggregation of the original equations. A computationally less demanding approximate reduction and linearization technique is also proposed.

1. Introduction

The past few years have witnessed a surge of interest in computational models based on ordinary differential equations (ODE's), ranging from continuous-time Markov chains (e.g. [4]) and process description languages oriented to bio-chemical systems (e.g. [42, 5, 13]), to deterministic approximations of stochastic systems (e.g. [18, 41]), and hybrid systems (e.g. [39, 37, 29]).

From a computational point of view, our motivation to study ODE's arises from the following problems.

(1) *Reasoning*: provide methods to automatically prove and discover identities involving the system variables.
(2) *Reduction*: provide methods to automatically reduce, and possibly minimize, the number of variables and equations of a system, in such a way that the reduced system retains all the relevant information of the original one.

Key words and phrases: Ordinary Differential Equations, Bisimulation, Minimization, Polynomials, Gröbner Bases.

* Extended and revised version of [9].

LOGICAL METHODS IN COMPUTER SCIENCE
DOI:10.23638/LMCS-15(1:14)2019

Reasoning may help an expert (a chemist, a biologist, an engineer) to prove or to disprove certain system properties, even before actually solving, simulating or realizing the system. Often, the identities of interest take the form of *conservation laws*. For instance, chemical reactions often enjoy a mass conservation law, stating that the sum of the concentrations of two or more chemical species, is a constant. More generally, one would like tools to automatically *discover* all laws of a given form. Pragmatically, before actually solving or simulating a given system, it can be critical to reduce the system to a size that can be handled by a solver or a simulator.

Our goal is showing that these issues can be dealt with by a mix of algebraic and coalgebraic techniques. We will consider *initial value* problems, specified by a system of ODE's of the form $\dot{x}_i = f_i(x_1, ..., x_N)$, for $i = 1, ..., N$, plus initial conditions. The functions f_is are called *drifts*; here we will focus on the case where the drifts are multivariate polynomials in the variables $x_1, .., x_N$. Practically, the majority of functions found in applications is in, or can be encoded into this format (possibly under restrictions on the initial conditions), including exponential, trigonometric, logarithmic and rational functions.

A more detailed account of our work follows. We introduce the ring of multivariate polynomials as a syntax for denoting the *behaviours* induced by the given initial value problem (Section 2). In other words, a behaviour is any polynomial combination of the individual components $x_i(t)$ $(i = 1, .., N)$ of the (unique) system solution. We then endow the polynomial ring with a transition system, based on a purely syntactic notion of *Lie derivative* (Section 3). This structure naturally induces a notion of bisimulation over polynomials, $\mathcal{L}$*-bisimilarity*, that is in agreement with the underlying ODE' s. In particular, any two variables x_i and x_j are $\mathcal{L}$-bisimilar if and only the corresponding solutions are the same, $x_i(t) = x_j(t)$ (this generalizes to polynomial behaviours as expected). This way, one can prove identities between two behaviours, for instance conservation laws, by exhibiting bisimulations containing the given pair . The resulting proof method is greatly enhanced by introducing a polynomial version of the *up to* technique of [36]. In order to turn this method into a fully automated proof procedure, we first characterize $\mathcal{L}$-bisimulation algebraically, in terms of certain *ideals* in the polynomial ring that are invariant under Lie-derivation (Section 4). This characterization leads to an algorithm that, given a user-specified template, returns the set of all its instances that are valid identities in the system (Section 5). One may use this algorithm, for instance, to discover all the conservation laws of the system involving terms up to a given degree. The algorithm implies building an ascending chain of ideals until stabilization, and relies on a few basic concepts from Algebraic Geometry, notably Gröbner bases [20]. The output of the algorithm is in turn essential to build a reduced system of ODE's, equivalent to the original one, but featuring a *minimal* number of equations and variables, in the class of systems that can be obtained by linear aggregation from the original one (Section 6). A computationally less demanding *approximate* reduction and linearization technique is proposed (Section 7). This may be an attractive alternative, because it is entirely based on linear algebraic, hence efficient, techniques, and produces 'small' linear systems. In particular, m linear equations are sufficient to guarantee an approximation within $O(t^m)$ of the original system. We then illustrate the results of some simple experiments we have conducted using a prototype implementation (in Python) of our algorithms (Section 8). Our approach is mostly related to some recent work on equivalences for ODE's by Cardelli et al. [14] and to work in the area of hybrid systems. We discuss this and other related work, as well as some possible directions for future work, in the concluding section (Section 9). In the

interest of readability, some proofs and technical material have been confined to a separate appendix (Appendix A).

To sum up, we give the following contributions.

(1) A complete bisimulation-based proof technique to reason on the polynomial behaviours induced by a system of ODE's.
(2) An algorithm to find all the valid polynomial identities induced by the given system and fitting a user-specified template.
(3) An algorithm to build a reduced, equivalent system that is minimal in the class of all linear aggregations of the original system.
(4) An algorithm to build a small linear system, approximating within $O(t^m)$ the original system.

2. Preliminaries

Let us fix an integer $N \geq 1$ and a set of N distinct variables $x_1, ..., x_N$. We will denote by $\mathbf{x}$ the column[1] vector $(x_1, ..., x_N)^T$. We let $\mathbb{R}[\mathbf{x}]$ denote the set of multivariate polynomials in the variables $x_1, ..., x_N$ with coefficients in $\mathbb{R}$, and let p, q range over it. Here we regard polynomials as syntactic objects. Given an integer $d \geq 0$, by $\mathbb{R}_d[\mathbf{x}]$ we denote the set of polynomials of degree $\leq d$. As an example, $p = 2xy^2 + (1/5)wz + yz + 1$ is a polynomial of degree $\deg(p) = 3$, that is $p \in \mathbb{R}_3[x, y, z, w]$, with monomials xy^2, wz, yz and 1. Depending on the context, with a slight abuse of notation it may be convenient to let a polynomial denote the induced function $\mathbb{R}^N \to \mathbb{R}$, defined as expected. In particular, x_i can be seen as denoting the projection on the i-th coordinate.

A (polynomial) *vector field* is a vector of N polynomials, $F = (f_1, ..., f_N)^T$, seen as a function $F : \mathbb{R}^N \to \mathbb{R}^N$. A vector field F and an initial condition $v_0 \in \mathbb{R}^N$ together define an *initial value problem* $\mathbf{\Phi} = (F, v_0)$, often written in the following form

$$\mathbf{\Phi} : \left\{ \begin{array}{lcl} \dot{\mathbf{x}}(t) & = & F(\mathbf{x}(t)) \\ \mathbf{x}(0) & = & v_0 \,. \end{array} \right. \tag{2.1}$$

The functions f_i in F are called *drifts* in this context. A *solution* to this problem is a differentiable function $\mathbf{x}(t) : D \to \mathbb{R}^N$, for some nonempty open interval $D \subseteq \mathbb{R}$ containing 0, which fulfills the above two equations, that is: $\frac{d}{dt}\mathbf{x}(t) = F(\mathbf{x}(t))$ for each $t \in D$ and $\mathbf{x}(0) = v_0$. By the Picard-Lindelöf theorem [2], there exists a nonempty open interval D containing 0, over which there is a *unique* solution, say $\mathbf{x}(t) = (x_1(t), ..., x_N(t))^T$, to the problem. In our case, as F is infinitely often differentiable, the solution is seen to be *analytic* in D: each $x_i(t)$ admits a Taylor series expansion in a neighborhood of 0. For definiteness, we will take the domain of definition D of $\mathbf{x}(t)$ to be the largest symmetric open interval where for each $i = 1, ..., N$, the Taylor expansion from 0 of $x_i(t)$ converges pointwise to $x_i(t)$ (possibly $D = \mathbb{R}$). The resulting vector function of t, $\mathbf{x}(t)$, is called the *time trajectory* of the system.

Given a differentiable function $g : E \to \mathbb{R}$, for some open set $E \subseteq \mathbb{R}^N$, the *Lie derivative of g along F* is the function $E \to \mathbb{R}$ defined as

$$\mathcal{L}_F(g) \triangleq \langle \nabla g, F \rangle = \sum_{i=1}^{N} (\frac{\partial g}{\partial x_i} \cdot f_i) \,.$$

[1] *Vector* means column vector, unless otherwise specified.

The Lie derivative of the sum $h+g$ and product $h \cdot g$ functions obey the familiar rules

$$\mathcal{L}_F(h+g) = \mathcal{L}_F(h) + \mathcal{L}_F(g) \tag{2.2}$$

$$\mathcal{L}_F(h \cdot g) = h \cdot \mathcal{L}_F(g) + \mathcal{L}_F(h) \cdot g\,. \tag{2.3}$$

Note that $\mathcal{L}_F(x_i) = f_i$. Moreover if $p \in \mathbb{R}_d[\mathbf{x}]$ then $\mathcal{L}_F(p) \in \mathbb{R}_{d+d'}[\mathbf{x}]$, for some integer $d' \geq 0$ that depends on d and on F. This allows us to view the Lie derivative of polynomials along a polynomial field F as a purely syntactic mechanism, that is as a function $\mathcal{L}_F : \mathbb{R}[\mathbf{x}] \to \mathbb{R}[\mathbf{x}]$ that does not assume anything about the solution of (2.1). Informally, we can view p as a program, and taking the Lie derivative of p can be interpreted as unfolding the definitions of the variables x_i's, according to the equations in (2.1) and to the formal rules for product and sum derivation, (2.2) and (2.3). We will pursue this view systematically in Section 3.

Example 2.1. Consider $N = 4$, $\mathbf{x} = (x, y, z, w)^T$ and the set of polynomials $\mathbb{R}[\mathbf{x}]$. The vector field $F = (xz + z, yw + z, z, w)^T$ and the initial condition $v_0 = (0, 0, 1, 1)^T$ together define an initial value problem (with no particular physical meaning) $\mathbf{\Phi} = (F, v_0)$. This problem can be equivalently written in the form

$$\begin{cases} \dot{x}(t) = x(t)z(t) + z(t) \\ \dot{y}(t) = y(t)w(t) + z(t) \\ \dot{z}(t) = z(t) \\ \dot{w}(t) = w(t) \\ \mathbf{x}(0) = v_0 = (0, 0, 1, 1)^T\,. \end{cases} \tag{2.4}$$

As an example of Lie derivative, if $p = 2xy^2 + wz$, we have $\mathcal{L}_F(p) = 4wxy^2 + 2wz + 2xy^2z + 4xyz + 2y^2z$.

The connection between time trajectories, polynomials and Lie derivatives can be summed up as follows. For any polynomial $p \in \mathbb{R}[\mathbf{x}]$, the function $p \circ \mathbf{x}(t) : D \to \mathbb{R}$, obtained by composing p as a function $\mathbb{R}^N \to \mathbb{R}$ with the time trajectory $\mathbf{x}(t) : D \to \mathbb{R}^N$, is analytic: we let $p(\mathbf{x}(t))$ denote the extension of this function over the largest symmetric open interval of convergence (possibly coinciding with $\mathbb{R}$) of its Taylor expansion from 0. We will call $p(\mathbf{x}(t))$ the *polynomial behaviour induced by p and by* the initial value problem (2.1). The connection between Lie derivatives of p along F and the initial value problem (2.1) is given by the following equations, which can be readily checked. Here and in the sequel, we let $p(v_0)$ denote the real number obtained by evaluating p at v_0.

$$p(\mathbf{x}(t))_{|t=0} = p(v_0) \tag{2.5}$$

$$\frac{d}{dt}p(\mathbf{x}(t)) = (\mathcal{L}_F(p))(\mathbf{x}(t))\,. \tag{2.6}$$

More generally, defining inductively $\mathcal{L}_F^{(0)}(p) \stackrel{\triangle}{=} p$ and $\mathcal{L}_F^{(j+1)}(p) \stackrel{\triangle}{=} \mathcal{L}_F(\mathcal{L}_F^j(p))$, we have the following equation for the j-th derivative of $p(\mathbf{x}(t))$ $(j = 0, 1, ...)$

$$\frac{d^j}{dt^j}p(\mathbf{x}(t)) = (\mathcal{L}_F^{(j)}(p))(\mathbf{x}(t))\,. \tag{2.7}$$

In the sequel, we shall often abbreviate $\mathcal{L}_F^{(j)}(p)$ as $p^{(j)}$, and shall omit the subscript $_F$ from $\mathcal{L}_F$ when clear from the context.

3. Coalgebraic semantics of polynomial ode's

In this section we show how to endow the polynomial ring with a transition relation structure, hence giving rise to coalgebra. Bisimilarity in this coalgebra will correspond to equality between polynomial behaviours.

We recall that a *(Moore) coalgebra* (see e.g. [34]) with outputs in a set O is a triple $C = (S, \delta, o)$ where S is a set of *states*, $\delta : S \to S$ is a *transition* function, and $o : S \to O$ is an *output* function. A *bisimulation* in C is a binary relation $R \subseteq S \times S$ such that whenever $s\,R\,t$ then: (a) $o(s) = o(t)$, and (b) $\delta(s)\,R\,\delta(t)$. It is an (easy) consequence of the general theory of bisimulation that a largest bisimulation over S, called bisimilarity and denoted by $\sim$, exists, is the union of all bisimulation relations, and is an equivalence relation over S.

Given an initial value problem $\mathbf{\Phi} = (F, v_0)$ of the form (2.1), the triple

$$C_{\mathbf{\Phi}} \stackrel{\triangle}{=} (\mathbb{R}[\mathbf{x}], \mathcal{L}_F, o)$$

forms a coalgebra with outputs in $\mathbb{R}$, where: (1) $\mathbb{R}[\mathbf{x}]$ is the set of states; (2) $\mathcal{L}_F$ acts as the transition function; and (3) o defined as $o(p) \stackrel{\triangle}{=} p(v_0)$ is the output function. Note that this definition of coalgebra is merely syntactic, and does not presuppose anything about the solution of the given initial value problem. When the standard definition of bisimulation over coalgebras is instantiated to $C_{\mathbf{\Phi}}$, it yields the following.

Definition 3.1 ($\mathcal{L}$-bisimulation $\sim_{\mathbf{\Phi}}$)**.** Let $\mathbf{\Phi}$ be an initial value problem. A binary relation $R \subseteq \mathbb{R}[\mathbf{x}] \times \mathbb{R}[\mathbf{x}]$ is a *$\mathcal{L}$-bisimulation* if, whenever $p\,R\,q$ then: (a) $p(v_0) = q(v_0)$, and (b) $\mathcal{L}(p)\,R\,\mathcal{L}(q)$. The largest $\mathcal{L}$-bisimulation over $\mathbb{R}[\mathbf{x}]$ is denoted by $\sim_{\mathbf{\Phi}}$.

We now introduce a new coalgebra with outputs in $\mathbb{R}$. Let $\mathcal{A}$ denote the family of real valued functions f such that f is analytic at 0 and f's domain of definition coincides with the open interval of convergence of its Taylor series (nonempty, centered at 0, possibly coinciding with $\mathbb{R}$)[2]. We define the coalgebra of analytic functions as

$$C_{\text{an}} \stackrel{\triangle}{=} (\mathcal{A}, (\cdot)', o_{\text{an}})$$

where $(f)' = \frac{df}{dt}$ is the standard derivative, and $o_{\text{fin}}(f) \stackrel{\triangle}{=} f(0)$ is the output function. We recall that a *morphism* μ between two coalgebras with outputs in the same set, $\mu : C_1 \to C_2$, is a function from states to states that preserves transitions ($\mu(\delta_1(s)) = \delta_2(\mu(s))$) and outputs ($o_1(s) = o_2(\mu(s))$). It is a standard (and easy) result of coalgebra that a morphism maps bisimilar states into bisimilar states: $s \sim_1 s'$ implies $\mu(s) \sim_2 \mu(s')$.

The coalgebra C_{an} has a special status, in that, given any coalgebra C with outputs in $\mathbb{R}$, *if* there is a morphism from C to C_{an}, this morphism is guaranteed to be *unique*. For our purposes, it is enough to focus on $C = C_{\mathbf{\Phi}}$. We define the function $\mu : \mathbb{R}[\mathbf{x}] \to \mathcal{A}$ as

$$\mu(p) \stackrel{\triangle}{=} p(\mathbf{x}(t))\,.$$

Theorem 3.2 (coinduction)**.** *μ is the unique morphism from $C_{\mathbf{\Phi}}$ to C_{an}. Moreover, the following* coinduction *principle is valid: $p \sim_{\mathbf{\Phi}} q$ in $C_{\mathbf{\Phi}}$ if and only if $p(\mathbf{x}(t)) = q(\mathbf{x}(t))$ in $\mathcal{A}$.*

Proof. The function μ given above is well defined, because $p(\mathbf{x}(t)) \in \mathcal{A}$, and is a morphism: for output and transition preservation, use (2.5) and (2.6), respectively. By the above recalled standard result in coalgebra, then $p \sim_{\mathbf{\Phi}} q$ implies $p(\mathbf{x}(t)) \sim q(\mathbf{x}(t))$ in C_{an}. Assume now that ν is a morphism from $C_{\mathbf{\Phi}}$ to C_{an}. From the definition of morphism and bisimulation,

[2]Equivalently, $\mathcal{A}$ is the set of power series $f(t) = \sum_{j\geq 0} a_j t^j$ with a positive radius of convergence.

it is readily checked that for each p, $\mu(p) \sim \nu(p)$ in C_{an}. Finally, we check that $\sim$ in C_{an} coincides with equality: indeed, if two functions are bisimilar in $\mathcal{A}$, then they have the same Taylor coefficients (this is shown by induction on the order of the derivatives, relying on the fact that $f \sim g$ means $f(0) = g(0)$ and $f' \sim g'$); the vice-versa is obvious. This completes the proof of both parts of the statement. □

Remark 3.3 (categorical presentation)**.** Existence of a morphism from any coalgebra with outputs in $\mathbb{R}$ into C_{an} is not guaranteed: for instance, the sequence (*stream*) of coefficients of any series with a radius of convergence 0, such as $(0!, 1!, 2!, ..., i!, ...)$, trivially induces a coalgebra from which no morphism to C_{an} exists. In this sense, C_{an} is not *final*. Note that C_{an} can be injected into the coalgebra of streams in the sense of Rutten [34], which is indeed final.

In a more categorical perspective, C_{an} induces a so-called *covariety* – see e.g. [22] – that is, the category of coalgebras that have a unique morphism into C_{an}: C_{an} is of course final in this covariety. How to characterise such covariety more explicitly, for example in terms of comonads, is left for future research.

Theorem 3.2 allows one to prove polynomial relations among the components $x_i(t)$ of $\mathbf{x}(t)$, say that $p(\mathbf{x}(t)) = q(\mathbf{x}(t))$, by coinduction, that is, by exhibiting a suitable $\mathcal{L}$-bisimulation relating the polynomials p and q.

Example 3.4. For $N = 2$, consider the vector field $F = (x_2, -x_1)^T$ with the initial value $v_0 = (0, 1)^T$. The binary relation $R \subseteq \mathbb{R}[x_1, x_2] \times \mathbb{R}[x_1, x_2]$ defined thus

$$R = \{\, (0, 0),\ (x_1^2 + x_2^2, 1) \,\}$$

is easily checked to be an $\mathcal{L}$-bisimulation. Thus we have proved the polynomial relation $x_1^2(t) + x_2^2(t) = 1$. Note that the unique solution to the given initial value problem is the pair of functions $\mathbf{x}(t) = (\sin(t), \cos(t))^T$. This way we have proven the familiar trigonometric identity $\sin(t)^2 + \cos(t)^2 = 1$.

This proof method can be greatly enhanced by a so called $\mathcal{L}$*-bisimulation up to* technique, in the spirit of [36]. This can be regarded as a form of up-to context technique, since we are just considering the closure w.r.t. the syntax, that is, the algebraic structure of the polynomial ring.

Definition 3.5 ($\mathcal{L}$-bisimulation up to)**.** Let $R \subseteq \mathbb{R}[\mathbf{x}] \times \mathbb{R}[\mathbf{x}]$ be a binary relation. Consider the binary relation $\widehat{R}$ defined by: $p\,\widehat{R}\,q$ iff there are $m \geq 0$ and polynomials h_i, p_i, q_i $(i = 1, ..., m)$ such that: $p = \sum_{i=1}^m h_i p_i$ and $q = \sum_{i=1}^m h_i q_i$ and $p_i\,R\,q_i$, for $i = 1, ..., m$.

A relation $R \subseteq \mathbb{R}[\mathbf{x}] \times \mathbb{R}[\mathbf{x}]$ is a $\mathcal{L}$*-bisimulation up to* if, whenever $p\,R\,q$ then: (a) $p(v_0) = q(v_0)$, and (b) $\mathcal{L}(p)\,\widehat{R}\,\mathcal{L}(q)$.

Note that, from the definition, for each relation R we have $R \subseteq \widehat{R}$.

Lemma 3.6. *Let R be an $\mathcal{L}$-bisimulation up to. Then $\widehat{R}$ is an $\mathcal{L}$-bisimulation, consequently $R \subseteq \sim_{\mathbf{\Phi}}$.*

Proof. In order to check that $\widehat{R}$ is an $\mathcal{L}$-bisimulation, assume that $p\,\widehat{R}\,q$, that is $p = \sum_{i=1}^m h_i p_i$ and $q = \sum_{i=1}^m h_i q_i$, for some h_i, p_i, q_i $(i = 1, ..., m)$ as specified by Definition 3.5. It is immediate to check that $p(v_0) = q(v_0)$, which proves condition (a) of the definition of $\mathcal{L}$-bisimulation. Furthermore, for each $i = 1, .., m$, we have by assumption that $\mathcal{L}(p_i)\,\widehat{R}\,\mathcal{L}(q_i)$.

That is, for suitable g_{ij}'s and r_{ij}'s, we have

$$\mathcal{L}(p_i) = \sum_j g_{ij} r_{ij} \qquad \mathcal{L}(q_i) = \sum_j g_{ij} s_{ij} \qquad r_{ij}\, R\, s_{ij}\,.$$

Recalling the rules for the Lie derivative, (2.2) and (2.3), we have

$$\begin{aligned}\mathcal{L}(p) = \sum_i h_i \mathcal{L}(p_i) + \mathcal{L}(h_i) p_i &= \sum_i h_i \sum_j g_{ij} r_{ij} + \mathcal{L}(h_i) p_i &= \sum_i \sum_j h_i g_{ij} r_{ij} + \sum_i \mathcal{L}(h_i) p_i \\ \widehat{R} \sum_i \sum_j h_i g_{ij} s_{ij} + \sum_i \mathcal{L}(h_i) q_i &= \sum_i h_i \sum_j g_{ij} s_{ij} + \sum_i \mathcal{L}(h_i) q_i &= \sum_i h_i \mathcal{L}(q_i) + \mathcal{L}(h_i) q_i \\ = \mathcal{L}(q)\ .\end{aligned}$$

This proves condition (b) of the definition of $\mathcal{L}$-bisimulation. □

Example 3.7. Consider the initial value problem of Example 2.1 and the relation defined below

$$R \;=\; \{\, (xz^j, yw^j)\,,\, (z^j, w^j) \;:\; j \geq 0\}\,.$$

It is easy to check that R is an $\mathcal{L}$-bisimulation up to. As an example, let us check condition (b) for a pair (xz^j, yw^j):

$$\mathcal{L}(xz^j) \;=\; (xz{+}z)z^j{+}jxz^j \;=\; (xz^{j+1}{+}jxz^j{+}zz^j)\;\widehat{R}\;(yw^{j+1}{+}jyw^j{+}zw^j) \;=\; (yw{+}z)w^j{+}jyw^j \;=\; \mathcal{L}(yw^j)\,.$$

This proves that $x(t) = y(t)$ and that $z(t) = w(t)$.

In the next two sections we will prove that this technique can be fully automated by resorting to the concept of *ideal* in a polynomial ring.

4. Algebraic characterization of $\mathcal{L}$-bisimilarity

We first review the notion of polynomial ideal from Algebraic Geometry, referring the reader to e.g. [20] for a comprehensive treatment. A set of polynomials $I \subseteq \mathbb{R}[\mathbf{x}]$ is an *ideal* if: (1) $0 \in I$, (2) I is closed under sum $+$, (3) I is absorbing under product $\cdot$, that is $p \in I$ implies $h \cdot p \in I$ for each $h \in \mathbb{R}[\mathbf{x}]$. Given a set of polynomials S, the ideal generated by S, denoted by $\left\langle\, S \,\right\rangle$, is defined as

$$\left\{ \sum_{j=1}^{m} h_j p_j \;:\; m \geq 0,\, h_j \in \mathbb{R}[\mathbf{x}] \text{ and } p_j \in S, \text{ for } j = 1, ..., m \right\}. \tag{4.1}$$

The polynomial coefficients h_j in the above definition are called *multipliers*. It is clear that $\left\langle\, S \,\right\rangle$ is the smallest ideal containing S, which implies that $\left\langle\, \left\langle\, S \,\right\rangle \,\right\rangle = \left\langle\, S \,\right\rangle$. Any set S such that $\left\langle\, S \,\right\rangle = I$ is called a *basis* of I. Every ideal in the polynomial ring $\mathbb{R}[\mathbf{x}]$ is finitely generated, that is has a finite basis (a version of Hilbert's basis theorem).

$\mathcal{L}$-bisimulations can be connected to certain types of ideals. This connection relies on Lie derivatives. First, we define the Lie derivative of any set $S \subseteq \mathbb{R}[\mathbf{x}]$ as follows

$$\mathcal{L}(S) \;\triangleq\; \{\mathcal{L}(p) : p \in S\}\,.$$

We say that S is a *pre-fixpoint* of $\mathcal{L}$ if $\mathcal{L}(S) \subseteq S$. $\mathcal{L}$-bisimulations can be characterized as particular pre-fixpoints of $\mathcal{L}$ that are also ideals, called *invariants*.

Definition 4.1 (invariant ideals)**.** Let $\mathbf{\Phi} = (F, v_0)$. An ideal I is a $\mathbf{\Phi}$-*invariant* if: (a) $p(v_0) = 0$ for each $p \in I$, and (b) I is a pre-fixpoint of $\mathcal{L}_F$.

We will drop the $\mathbf{\Phi}$- from $\mathbf{\Phi}$-*invariant* whenever this is clear from the context. The following definition and lemma provide the link between invariants and $\mathcal{L}$-bisimulation.

Definition 4.2 (kernel)**.** The *kernel* of a binary relation $R \subseteq \mathbb{R}[\mathbf{x}] \times \mathbb{R}[\mathbf{x}]$ is $\ker(R) \stackrel{\triangle}{=} \{p - q : p \, R \, q\}$.

Lemma 4.3. *Let R be a binary relation. If R is an $\mathcal{L}$-bisimulation then $\left\langle\ \ker(R)\ \right\rangle$ is an invariant. Conversely, given an invariant I, then $R = \{(p, q) : p - q \in I\}$ is an $\mathcal{L}$-bisimulation.*

Consequently, proving that $p \sim_{\mathbf{\Phi}} q$ is equivalent to exhibiting an invariant I such that $p - q \in I$.

Example 4.4. Consider the initial value problem of Example 3.7. Let $I = \left\langle\ \{x-y, z-w\}\ \right\rangle$. Let us check that I is an invariant. Let $p = h_1(x - y) + h_2(z - w)$ be a generic element of I. Clearly $p(v_0) = 0$, thus condition (a) is satisfied. Concerning (b), we consider the two summands separately:

$$\begin{aligned}
\mathcal{L}(h_1(x-y)) &= \mathcal{L}(h_1)(x-y) + h_1\mathcal{L}(x-y) \\
&= \mathcal{L}(h_1)(x-y) + h_1(xz + z - (yz + w)) \\
&= \big(\mathcal{L}(h_1) + h_1 z\big)(x-y) + h_1(z-w) \\
&\in I \\
\mathcal{L}(h_2(z-w)) &= \mathcal{L}(h_1)(z-w) + h_1\mathcal{L}(z-w) \\
&= \mathcal{L}(h_1)(z-w) + h_1(z-w) \\
&= \big(\mathcal{L}(h_1) + h_1\big)(z-w) \\
&\in I\,.
\end{aligned}$$

Consequently, $\mathcal{L}(p) = \mathcal{L}(h_1(x - y)) + \mathcal{L}(h_2(z - w)) \in I$.

A more general problem than equivalence checking is finding *all* valid polynomial equations of a given form. We will illustrate an algorithm to this purpose in the next section.

The following result sums up the different characterization of $\mathcal{L}$-bisimilarity $\sim_{\mathbf{\Phi}}$. In what follows, we will denote the constant zero function in $\mathcal{A}$ simply by 0 and consider the following set of polynomials.

$$\mathcal{Z}_{\mathbf{\Phi}} \stackrel{\triangle}{=} \{p : p(\mathbf{x}(t)) \text{ is identically } 0\}\,.$$

The following result also proves that $\mathcal{Z}_{\mathbf{\Phi}}$ is the largest $\mathbf{\Phi}$-invariant.

Theorem 4.5 ($\mathcal{L}$-bisimilarity via ideals)**.** *We have the following characterizations of $\mathcal{L}$-bisimilarity. For any pair of polynomials p and q:*

$$\begin{aligned}
p \sim_{\mathbf{\Phi}} q \ \textit{iff}\ p - q &\in \ker(\sim_{\mathbf{\Phi}}) && (4.2)\\
&= \mathcal{Z}_{\mathbf{\Phi}} && (4.3)\\
&= \left\{p : p^{(j)}(v_0) = 0 \textit{ for each } j \geq 0\right\} && (4.4)\\
&= \bigcup \{I : I \textit{ is a } \mathbf{\Phi}\textit{-invariant}\}\,. && (4.5)
\end{aligned}$$

5. Computing invariants

By Theorem 4.5, proving $p \sim_{\mathbf{\Phi}} q$ means finding an invariant I such that $p - q \in I \subseteq \mathcal{Z}_{\mathbf{\Phi}}$. More generally, we focus here on the problem of finding invariants that include a user-specified set of polynomials. In the sequel, we will make use of the following two basic facts about ideals, for whose proof we refer the reader to [20].

(1) Any infinite ascending chain of ideals in a polynomial ring, $I_0 \subseteq I_1 \subseteq \cdots$, stabilizes at some finite k. That is, there is $k \geq 0$ such that $I_k = I_{k+j}$ for each $j \geq 0$. This is just another version of Hilbert's basis theorem.
(2) The *ideal membership problem*, that is, deciding whether $p \in I$, given p and a finite set of S of *generators* (such that $I = \langle\ S\ \rangle$), is decidable (provided the coefficients used in p and in S can be finitely represented), although it requires exponential space in the number of variables. The ideal membership will be further discussed later on in the section.

The main idea is introduced by the naive algorithm presented below.

5.1. **A naive algorithm.** Suppose we want to decide whether $p \in \mathcal{Z}_{\mathbf{\Phi}}$. It is quite easy to devise an algorithm that computes the smallest invariant containing p, or returns 'no' in case no such invariant exists, i.e. in case $p \notin \mathcal{Z}_{\mathbf{\Phi}}$. Consider the successive Lie derivatives of p, $p^{(j)} = \mathcal{L}^{(j)}(p)$ for $j = 0, 1, ...$ For each $j \geq 0$, let $I_j \stackrel{\triangle}{=} \langle\ \{p^{(0)}, ..., p^{(j)}\}\ \rangle$. Let m be the least integer such that either

(a) $p^{(m)}(v_0) \neq 0$, or
(b) $I_m = I_{m+1}$.

If (a) occurs, then $p \notin \mathcal{Z}_{\mathbf{\Phi}}$, so we return 'no' (Theorem 4.5(4.4)); if (b) occurs, then I_m is the least invariant containing p. Note that the integer m is well defined: $I_0 \subseteq I_1 \subseteq I_2 \subseteq \cdots$ forms an infinite ascending chain of ideals, which must stabilize in a finite number of steps (fact 1 at the beginning of the section). In particular, as soon as $I_{i+1} = I_i$ the chain gets stable, as a consequence of the derivation rules (2.2) and (2.3).

Checking condition (b) amounts to deciding if $p^{(m+1)} \in I_m$. This is an instance of the ideal membership problem, which can be solved effectively. Generally speaking, given a polynomial p and finite set of polynomials S, deciding the ideal membership $p \in I = \langle\ S\ \rangle$ can be accomplished by first transforming S into a *Gröbner basis* G for I (via, e.g. the Buchberger's algorithm), then computing r, the *residual* of p modulo G (via a sort generalised division of p by G): one has that $p \in I$ if and only if $r = 0$ (again, this procedure can be carried out effectively only if the coefficients involved in p and S are finitely representable; in practice, one often confines to rational coefficients). We refer the reader to [20] for further details on the ideal membership problem. Known procedures to compute Gröbner bases have exponential worst-case space complexity depending on the number of variables, although may perform reasonably well in some concrete cases. One should in any case invoke such procedures parsimoniously. Here is a small example to illustrate the above outlined algorithm.

Example 5.1. Consider again the initial value problem of Example 3.7. Let $p = x - y$. With the help of a computer algebra system, we can easily check the following.

- $p^{(0)} = p$ and $p^{(0)}(v_0) = 0$;
- $p^{(1)} = xz - yw$, $p^{(1)}(v_0) = 0$ and $p^{(1)} \notin I_0 = \langle\ \{p^{(0)}\}\ \rangle$;

- $p^{(2)} = -w^2y - wy - wz + xz^2 + xz + z^2$, $p^{(2)}(v_0) = 0$ and $p^{(2)} \notin I_1 = \langle\ \{p^{(0)}, p^{(1)}\}\ \rangle$
- $p^{(3)} = -w^3y - 3w^2y - w^2z - wy - 3wz + xz^3 + 3xz^2 + xz + z^3 + 3z^2$, $p^{(3)}(v_0) = 0$ and finally[3] $p^{(3)} \in I_2 = \langle\ \{p^{(0)}, p^{(1)}, p^{(2)}\}\ \rangle$.

Hence I_2 is the least invariant containing $p = x - y$, thus proving that $x - y \in \mathcal{Z}_{\mathbf{\Phi}}$.

We will introduce below a more general algorithm, which can also deal with (infinite) *sets* of user-specified polynomials. First, we need to introduce the concept of template.

5.2. **Templates.** Polynomial templates have been introduced by Sankaranarayanan, Sipma and Manna in [37] as a means to compactly specify sets of polynomials. Fix a tuple of $n \geq 1$ of distinct *parameters*, say $\mathbf{a} = (a_1, ..., a_n)$, disjoint from $\mathbf{x}$. Let $Lin(\mathbf{a})$, ranged over by ℓ, be the set of *linear expressions* with coefficients in $\mathbb{R}$ and variables in $\mathbf{a}$; e.g. $\ell = 5a_1 + 42a_2 - 3a_3$ is one such expression[4]. A *template* is a polynomial in $Lin(\mathbf{a})[\mathbf{x}]$, that is, a polynomial with linear expressions as coefficients; we let π range over templates. For example, the following is a template:

$$\pi \quad = \quad (5a_1 + (3/4)a_3)xy^2 + (7a_1 + (1/5)a_2)xz + (a_2 + 42a_3)\,.$$

Given a vector $v = (\lambda_1, ..., \lambda_n)^T \in \mathbb{R}^n$, we will let $\ell[v] \in \mathbb{R}$ denote the result of replacing each parameter a_i with λ_i, and evaluating the resulting expression; we will let $\pi[v] \in \mathbb{R}[\mathbf{x}]$ denote the polynomial obtained by replacing each ℓ with $\ell[v]$ in π. Given a set $S \subseteq \mathbb{R}^n$, we let $\pi[S]$ denote the set $\{\pi[v] : v \in S\} \subseteq \mathbb{R}[\mathbf{x}]$.

The (formal) Lie derivative of π is defined as expected, once linear expressions are treated as constants; note that $\mathcal{L}(\pi)$ is still a template. It is easy to see that the following property is true: for each π and v, one has $\mathcal{L}(\pi[v]) = \mathcal{L}(\pi)[v]$. This property extends as expected to the j-th Lie derivative ($j \geq 0$)

$$\mathcal{L}^{(j)}(\pi[v]) \quad = \quad \mathcal{L}^{(j)}(\pi)[v]\,. \tag{5.1}$$

5.3. **A double chain algorithm.** We present an algorithm that, given a template π with n parameters, returns a pair (V, J), where $V \subseteq \mathbb{R}^n$ is such that $\pi[\mathbb{R}^n] \cap \mathcal{Z}_{\mathbf{\Phi}} = \pi[V]$, and J is the smallest invariant that includes $\pi[V]$. We first give a purely mathematical description of the algorithm, postponing its effective representation to the next subsection. The algorithm is based on building two chains of sets, a descending chain of vector spaces and an (eventually) ascending chain of ideals. The ideal chain is used to detect the stabilization of the sequence. In fact, in the sequence of vector spaces below, $V_{i+1} = V_i$ does not in itself imply that $V_{i+k} = V_i$ for each $k \geq 1$. Formally, for each $i \geq 0$, consider the sets

$$V_i \quad \triangleq \quad \{v \in \mathbb{R}^n \,:\, \pi^{(j)}[v](v_0) = 0 \text{ for } j = 0, ..., i\,\} \tag{5.2}$$

$$J_i \quad \triangleq \quad \langle\ \bigcup_{j=0}^{i} \pi^{(j)}[V_i]\ \rangle\,. \tag{5.3}$$

[3]Indeed, a Gröbner basis for I_2 is $G = \{x - y, yz - wy, z^2 - wz\}$, and $p^{(3)} = (z^3 + 3z^2 + z)(x - y) + (w^2 + wz + 3w + z^2 + 3z + 1)(yz - wy) + (w + z + 3)(z^2 - wz)$.

[4]Differently from Sankaranarayanan et al. we do not allow linear expressions with a constant term, such as $2 + 5a_1 + 42a_2 - 3a_3$. This minor syntactic restriction does not practically affect the expressiveness of the resulting polynomial templates.

It is easy to check that each $V_i \subseteq \mathbb{R}^n$ is a vector space over $\mathbb{R}$ of dimension $\leq n$. Now let $m \geq 0$ be the least integer such that the following conditions are *both* true:

$$V_{m+1} = V_m \tag{5.4}$$
$$J_{m+1} = J_m\,. \tag{5.5}$$

The algorithm returns (V_m, J_m). Note that the integer m is well defined: indeed, $V_0 \supseteq V_1 \supseteq \cdots$ forms an infinite descending chain of finite-dimensional vector spaces, which must stabilize in finitely many steps. In other words, we can consider the least m' such that $V_{m'} = V_{m'+k}$ for each $k \geq 1$. Then $J_{m'} \subseteq J_{m'+1} \subseteq \cdots$ forms an infinite ascending chain of ideals, which must stabilize at some $m \geq m'$. Therefore there must be some index m such that (5.4) and (5.5) are both satisfied, and we choose the least such m.

The next theorem states the correctness and relative completeness of this abstract algorithm. Informally, the algorithm will output the largest space V_m such that $\pi[V_m] \subseteq \mathcal{Z}_{\mathbf{\Phi}}$ and the smallest invariant J_m witnessing this inclusion. Note that, while typically the user will be interested in $\pi[V_m]$, J_m as well may contain useful information, such as higher order, nonlinear conservation laws. We need a technical lemma.

Lemma 5.2. *Let V_m, J_m be the sets returned by the algorithm. Then for each $j \geq 1$, one has $V_m = V_{m+j}$ and $J_m = J_{m+j}$.*

Theorem 5.3 (correctness and relative completeness)**.** *Let V_m, J_m be the sets returned by the algorithm for a polynomial template π.*
(a) $\pi[V_m] = \mathcal{Z}_{\mathbf{\Phi}} \cap \pi[\mathbb{R}^n]$;
(b) J_m is the smallest invariant containing $\pi[V_m]$.

Proof. Concerning part (a), we first note that $\pi[v] \in \mathcal{Z}_{\mathbf{\Phi}} \cap \pi[\mathbb{R}^n]$ means $(\pi[v])^{(j)}(v_0) = \pi^{(j)}[v](v_0) = 0$ for each $j \geq 0$ (Theorem 4.5(4.4)), which, by definition, implies $v \in V_j$ for each $j \geq 0$, hence $v \in V_m$. Conversely, if $v \in V_m = V_{m+1} = V_{m+2} = \cdots$ (here we are using Lemma 5.2), then by definition $(\pi[v])^{(j)}(v_0) = \pi^{(j)}[v](v_0) = 0$ for each $j \geq 0$, which implies that $\pi[v] \in \mathcal{Z}_{\mathbf{\Phi}}$ (again Theorem 4.5(4.4)). Note that in proving both inclusions we have used property (5.1).

Concerning part (b), it is enough to prove that: (1) J_m is an invariant, (2) $J_m \supseteq \mathcal{Z}_{\mathbf{\Phi}} \cap \pi[\mathbb{R}^n]$, and (3) for any invariant I such that $\mathcal{Z}_{\mathbf{\Phi}} \cap \pi[\mathbb{R}^n] \subseteq I$, we have that $J_m \subseteq I$. We first prove (1), that J_m is an invariant. Indeed, for each $v \in V_m$ and each $j = 0, ..., m-1$, we have $\mathcal{L}(\pi^{(j)}[v]) = \pi^{(j+1)}[v] \in J_m$ by definition, while for $j = m$, since $v \in V_m = V_{m+1}$, we have $\mathcal{L}(\pi^{(m)}[v]) = \pi^{(m+1)}[v] \in J_{m+1} = J_m$ (note that in both cases we have used property (5.1)). Concerning (2), note that $J_m \supseteq \pi[V_m] = \mathcal{Z}_{\mathbf{\Phi}} \cap \pi[\mathbb{R}^n]$ by virtue of part (a). Concerning (3), consider any invariant $I \supseteq \mathcal{Z}_{\mathbf{\Phi}} \cap \pi[\mathbb{R}^n]$. We show by induction on $j = 0, 1, ...$ that for each $v \in V_m$, $\pi^{(j)}[v] \in I$; this will imply the wanted statement. Indeed, $\pi^{(0)}[v] = \pi[v] \in \mathcal{Z}_{\mathbf{\Phi}} \cap \pi[\mathbb{R}^n]$, as $\pi[V_m] \subseteq \mathcal{Z}_{\mathbf{\Phi}}$ by (a). Assuming now that $\pi^{(j)}[v] \in I$, by invariance of I we have $\pi^{(j+1)}[v] = \mathcal{L}(\pi^{(j)}[v]) \in I$ (again, we have used here property (5.1)). □

According to Theorem 5.3(a), given a template π and $v \in \mathbb{R}^n$, checking if $\pi[v] \in \pi[V_m]$ is equivalent to checking if $v \in V_m$, which can be effectively done knowing a basis B_m of V_m. We show how to effectively compute such a basis in the following.

5.4. **Effective representation.** For $i = 0, 1, ...$, we have to give effective ways to:
(i) represent the sets V_i, J_i in (5.2) and (5.3); and
(ii) check the termination conditions (5.4) and (5.5).

It is quite easy to address (i) and (ii) in the case of the vector spaces V_i. For each i, consider the linear expression $\pi^{(i)}(v_0)$. By factoring out the parameters $a_1, ..., a_n$ in this expression, we can write, for a suitable (row) vector of coefficients $t_i = (t_{i1}, ..., t_{in}) \in \mathbb{R}^{1\times n}$:

$$\pi^{(i)}(v_0) \quad = \quad t_{i1} \cdot a_1 + \cdots + t_{in} \cdot a_n$$

The condition on $v \in \mathbb{R}^n$, $\pi^{(i)}[v](v_0) = 0$, can then be translated into the linear constraint on v

$$t_i \cdot v \quad = \quad 0\,. \tag{5.6}$$

Letting $T_i \in \mathbb{R}^{i\times n}$ denote the matrix obtained by stacking the rows $t_1, ..., t_i$ on above the other, we see that V_i is the right null space of T_i. That is (here, $\mathbf{0}_i$ denotes the null vector in $\mathbb{R}^i$):

$$V_i \quad = \quad \{v \in \mathbb{R}^n \,:\, T_i v = \mathbf{0}_i\}\,.$$

Checking whether $V_i = V_{i+1}$ or not amounts then to checking whether the vector t_{i+1} is or not linearly dependent from the rows in T_i, which can be accomplished by standard and efficient linear algebraic techniques. In practice, the linear constraints (5.6) can be resolved and propagated incrementally[5], as they are generated, following the computation of the derivatives $\pi^{(i)}$. Concerning the representation of the ideals J_i, we will use the following lemma[6].

Lemma 5.4. *Let $V \subseteq \mathbb{R}^n$ be a vector space with B as a basis, and $\pi_1, ..., \pi_k$ be templates. Then $\langle\ \cup_{j=0}^k \pi_j[V]\ \rangle = \langle\ \cup_{j=0}^k \pi_j[B]\ \rangle$.*

Now let B_i be a finite basis of V_i, which can be easily built from the matrix T_i. By the previous lemma, $\cup_{j=1}^i \pi^{(j)}[B_i]$ is a *finite* set of generators for J_i: this solves the representation problem. Concerning the termination condition, we note that, after checking that actually $V_i = V_{i+1}$, checking $J_i = J_{i+1}$ reduces to checking that

$$\pi^{(i+1)}[B_i] \quad \subseteq \quad \langle\ \cup_{j=0}^i \pi^{(j)}[B_i]\ \rangle \ = \ J_i\,. \tag{5.7}$$

To check this inclusion, one can apply standard computer algebra techniques. For example, one can check if $\pi^{(i+1)}[b] \in J_i$ for each $b \in B_i$, thus solving $|B_i|$ ideal membership problems, for one and the same ideal J_i. As already discussed, this presupposes the computation of a Gröbner basis for J_i, a potentially expensive operation. One advantage of the above algorithm, over methods proposed in program analysis with termination conditions based on testing ideal membership (e.g. [32]), is that (5.7) is not checked at every iteration, but only when $V_{i+1} = V_i$ (the latter a relatively inexpensive check). In the Appendix (Subsection A.2), we also discuss a sufficient condition which does not involve Gröbner bases, but leads to an incomplete algorithm.

[5]E.g., if $\pi = a_1 x + a_2 y + a_3 x + a_4 w$ and $v_0 = (0, 0, 1, 1)^T$, then $\pi[v](v_0) = 0$ is resolved and propagated applying to π the substitution $[a_3 \mapsto -a_4]$.

[6]The restriction that linear expressions in templates do not contain constant terms is crucial here.

Example 5.5. Consider the initial value problem of Example 2.1 and the template $\pi = a_1x + a_2y + a_3z + a_4w$. We run the double chain algorithm with this system and template as inputs. In what follows, $v = (v_1, v_2, v_3, v_4)^T$ will denote a generic vector in $\mathbb{R}^4$. Recall that $\mathbf{x} = (x, y, z, w)^T$ and $v_0 = (0, 0, 1, 1)^T$.

- For each $v \in \mathbb{R}^4$: $\pi^{(0)}[v](v_0) = (v_1x + v_2y + v_3z + v_4w)(v_0) = 0$ if and only if $v \in V_0 \stackrel{\triangle}{=} \{v : v_3 = -v_4\}$.
- For each $v \in V_0$: $\pi^{(1)}[v](v_0) = (v_1xz + v_1z + v_2wy + v_2z + v_4w - v_4z)(v_0) = 0$ if and only if $v \in V_1 \stackrel{\triangle}{=} \{v \in V_0 : v_1 = -v_2\}$.
- For each $v \in V_1$: $\pi^{(2)}[v](v_0) = (v_2w^2y + v_2wy + v_2wz - v_2xz^2 - v_2xz - v_2z^2 + v_4w - v_4z)(v_0) = 0$ if and only if $v \in V_2 \stackrel{\triangle}{=} V_1$.
  Being $V_2 = V_1$, we also check if $J_2 = J_1$. A basis of V_1 is $B_1 = \{b_1, b_2\}$ with $b_1 = (-1, 1, 0, 0)^T$ and $b_2 = (0, 0, -1, 1)^T$. According to (5.7), we have therefore to check if, for $\ell = 1, 2$:

$$\pi^{(2)}[b_\ell] \quad \in \quad J_1 \stackrel{\triangle}{=} \left\langle\, \{\pi^{(0)}[b_1], \pi^{(0)}[b_2], \pi^{(1)}[b_1], \pi^{(1)}[b_2]\} \,\right\rangle.$$

  With the help of a computer algebra system, one computes a Gröbner basis for J_1 as $G_1 = \{x - y, z - w\}$. Then one can reduce $\pi^{(2)}[b_1] = w^2y + wy + wz - xz^2 - xz - z^2$ modulo G_1 and obtain $\pi^{(2)}[b_1] = h_1(x - y) + h_2(z - w)$, with $h_1 = -z^2 - z$ and $h_2 = -wy - yz - y - z$, thus proving that $\pi^{(2)}[b_1] \in J_1$. One proves similarly that $\pi^{(2)}[b_2] \in J_1$. This shows that $J_2 = J_1$.

Hence the algorithm terminates with $m = 1$ and returns (V_1, J_1), or, concretely, (B_1, G_1). In particular, $x - y \in \mathcal{Z}_{\mathbf{\Phi}}$, or equivalently $x(t) = y(t)$. Similarly for $z - w$.

Remark 5.6 (notational convention: result template)**.** According to Theorem 5.3(a), given a template π and $v \in \mathbb{R}^n$, checking if $\pi[v] \in \pi[V_m]$ is equivalent to checking if $v \in V_m$, which can be effectively done knowing the basis B_m of V_m concretely returned by the algorithm (another, equivalent possibility, is checking if v is orthogonal to the space $V_m^\perp$, which is built in the minimization phase, see next section).

In practice, it is more convenient to represent the whole set $\pi[V]$ returned by the algorithm more compactly, in terms of a *new* m-parameters result template π' such that $\pi'[\mathbb{R}^m] = \pi[V]$. For instance, in the previous example, the result template

$$\pi' \quad = \quad a_1x - a_1y + a_2z - a_2w$$

represents the algorithm's outcome $\pi[V_2]$, in the precise sense that $\pi[V_2] = \pi'[\mathbb{R}^2]$.

Remark 5.7 (linear systems)**.** Consider the case of a *linear* system, that is, when the drifts f_i in F are linear functions of the x_i's. Consider the chain $V_0 \supseteq V_1 \supseteq \cdots$ in (5.2). It is easy to prove that as soon as $V_{m+1} = V_m$ then the chain has stabilized, that is $V_{m+k} = V_m$ for each $k \geq 0$. Therefore, for linear systems, stabilization can be detected without looking at the ideals chain (5.3), hence dispensing with Gröbner bases. The resulting single chain algorithm boils down to the 'refinement' algorithm of [7, Th.2].

Remark 5.8. We end this section pointing out that the naive algorithm of Subsection 5.1 admits a generalization that works with templates as well. Specifically, one can regard templates as polynomials in both the variables $\mathbf{x}$ and parameters $\mathbf{a}$, and compute the invariant ideal in $\mathbb{R}[\mathbf{x}, \mathbf{a}]$ generated by all Lie derivatives of the given template π. Having computed this, one substitutes v_0 for $\mathbf{x}$. This procedure is related to what is done by Müller-Olm and Seidl in the discrete-time case [26] (further coniderations in the concluding section).

In any case, the idea of the algorithm in Subsection 5.2 is to avoid treating the parameters in the template *symbolically*— which is important, bearing in mind that the ideal membership problem requires exponential space in the number of variables.

6. Minimization

We present a method for reducing the size of an initial value problem. The resulting reduced problem, while equivalent in a precise sense to the original problem, is *minimal* in terms of number of equations and variables, among all systems that can be obtained by linear aggregations of the original equations.

The method takes as an input the space V_m returned by the double chain algorithm in the preceding section when fed with a certain linear template. The method itself is quite simple and only relies on simple linear algebraic operations that can be efficiently automated.

The basic idea is projecting the original system of equations onto a suitably chosen subspace of $\mathbb{R}^N$. Consider the linear template

$$\pi \quad = \quad a_1 \cdot x_1 + \cdots + a_N \cdot x_N \tag{6.1}$$

where the a_i's are distinct parameters. By applying the algorithm of the preceding section to this template, we obtain a subspace $V \stackrel{\triangle}{=} V_m \subseteq \mathbb{R}^N$. Consider now the orthogonal complement of V in $\mathbb{R}^N$ (where $\langle \cdot, \cdot \rangle$ is the usual scalar product between vectors in $\mathbb{R}^N$ and v, w denote generic vectors in $\mathbb{R}^N$)

$$W \quad \stackrel{\triangle}{=} \quad V^{\perp} \; = \; \{ w \in \mathbb{R}^N \; : \; \langle w, v \rangle = 0 \text{ for each } v \in V \} \, .$$

We show that the trajectory $\mathbf{x}(t)$ lies entirely in W, that is $\mathbf{x}(t) \in W$ for each t in the open interval of definition, say D, of the trajectory. Indeed, by virtue of Theorem 5.3(a), we have that $v \in V$ if and only if (here $v = (\lambda_1, ..., \lambda_N)^T$)

$$\langle \mathbf{x}, v \rangle \quad = \quad \textstyle\sum_{i=1}^{N} \lambda_i x_i \quad = \quad \pi[v] \quad \in \quad \mathcal{Z}_{\mathbf{\Phi}} \, .$$

By definition of $\mathcal{Z}_{\mathbf{\Phi}}$, this means that $v \in V$ if and only if

$$\langle \mathbf{x}(t), v \rangle \quad = \quad \textstyle\sum_{i=1}^{N} \lambda_i x_i(t) \quad = \quad \pi[v](t) \quad = \quad 0 \quad \text{for each } t$$

that is $\mathbf{x}(t) \in W$ for each t in the open interval D of definition of $\mathbf{x}(t)$. In other words, $v \in V$ if and only if v is orthogonal to each $\mathbf{x}(t)$, for $t \in D$, or $V = \{ \mathbf{x}(t) : t \in D \}^{\perp}$. This is equivalent to the following crucial lemma.

Lemma 6.1. $W = V^{\perp} = \text{span}\{ \, \mathbf{x}(t) : t \in D \, \}$.

The fact that the trajectory $\mathbf{x}(t)$ entirely lies in, and in fact generates, the subspace W, suggests that we can obtain a more economical representation of $\mathbf{x}(t)$ by adopting a system of coordinates in this subspace. More formally, let B be any orthonormal basis of W. It is convenient to represent B as a matrix of l independent column vectors, $B = [b_1 | \cdots | b_l] \in \mathbb{R}^{N \times l}$, where $l \stackrel{\triangle}{=} \dim(W) \leq N$ (in fact, $l \leq m + 1$ as well; this is discussed in the Appendix, Subsection A.3, where an efficient method for building B out of the successive Lie derivatives of π is explained). Recall that orthonormality means $B^T B = I_l$, with I_l the $l \times l$ identity matrix. The orthogonal projection of any v onto W has, w.r.t. B, coordinates $B^T v$, which is of course a vector in $\mathbb{R}^l$. Define now

$$\mathbf{y}(t) \quad \stackrel{\triangle}{=} \quad B^T \mathbf{x}(t) \quad \text{for each } t \in D \, . \tag{6.2}$$

Since each $\mathbf{x}(t) \in W$, each $\mathbf{x}(t)$ is a fixpoint of the projection, so with the above definition we have

$$\begin{aligned} \mathbf{x}(t) &= BB^T\mathbf{x}(t) \\ &= B\mathbf{y}(t)\,. \end{aligned} \tag{6.3}$$

From the last equation, it is easy to check that $\mathbf{y}(t)$ is a solution, hence the *unique* analytic solution in a suitable interval, of the following *reduced* problem $\mathbf{\Psi} = (G, \mathbf{y}(0))$, where F denotes the vector field of the original initial value problem:

$$\mathbf{\Psi}: \left\{ \begin{array}{rcl} \dot{\mathbf{y}}(t) &=& B^T F\left(B\mathbf{y}(t)\right) \\ \mathbf{y}(0) &=& B^T\mathbf{x}(0)\,. \end{array} \right. \tag{6.4}$$

In order to check that $\mathbf{y}(t)$ as defined in (6.2) satisfies the first equation of (6.4), observe that, by definition we have:

$$\begin{aligned} \dot{\mathbf{y}}(t) &= B^T\dot{\mathbf{x}}(t) \\ &= B^T F\left(\mathbf{x}(t)\right) \\ &= B^T F\left(B\mathbf{y}(t)\right) \end{aligned}$$

where the last equality follows from (6.3). The second equality of (6.4) is trivially seen to be true. Note that the reduced system (6.4) features $l \leq N$ differential equations. In particular, observe that the vector field G of the reduced system is obtained by replacing each variable x_i in the original F with a linear combination of the variables y_j' s, as dictated by B, and then linearly aggregating the resulting N terms, as dictated by B^T. As a consequence, the maximum degree in the reduced G does not exceed the maximum degree in the original F.

Equation (6.3) naturally extends to any polynomial behaviour. That is, for any polynomial $p \in \mathbb{R}[\mathbf{x}]$, we have $p(\mathbf{x}(t)) = p\left(\mathbf{x}(t)\right) = p\left(B\mathbf{y}(t)\right)$. In the end, we have proven the following result, which shows that we can exactly recover any behaviour induced by the original system from the reduced system.

Theorem 6.2 (exact reduction)**.** *Let $\mathbf{y}(t)$ be the unique analytic solution of the (reduced) problem* (6.4)*. Then, $\mathbf{x}(t) = B\mathbf{y}(t)$. Moreover, for any polynomial behaviour $p(\mathbf{x}(t))$ induced by the original problem* (2.1)*, we have $p(\mathbf{x}(t)) = p\left(B\mathbf{y}(t)\right)$.*

The reduced system is minimal among all systems obtained as linear aggregations of the original system. In the next result, the interpretation of the k-dimensional vector function $\mathbf{z}(t)$ is that it may (but need not) arise as the solution of any system with k equations.

Theorem 6.3 (minimality)**.** *Assume for some $N \times k$ matrix C and vector function $\mathbf{z}(t)$, we have $\mathbf{x}(t) = C\mathbf{z}(t)$, for each $t \in D$. Then $k \geq l$.*

Proof. Assume $k \leq N$ (otherwise there is nothing to prove). As the trajectory $\mathbf{x}(t)$ spans W (Lemma 6.1), which has dimension l, we can form a rank l matrix E as $E = [\mathbf{x}(t_1)|\cdots|\mathbf{x}(t_l)] = [C\mathbf{z}(t_1)|\cdots|C\mathbf{z}(t_l)] = C[\mathbf{z}(t_1)|\cdots|\mathbf{z}(t_l)]$, for suitable points $t_1, ..., t_l$. As in general $\mathrm{rk}(AB) \leq \min\{\mathrm{rk}(A), \mathrm{rk}(B)\}$, we have $\mathrm{rk}(C) \geq l$. But $k \geq \mathrm{rk}(C)$, which implies the thesis. □

As a corollary of Theorem 6.2, we obtain a further characterization of $\mathcal{L}$-bisimilarity, which shows that we can also reason syntactically on polynomial behaviours in terms of the reduced system. In particular, $\mathcal{L}$-bisimilarity between pairs of individual variables reduces to plain equality between the corresponding rows of B, which allows one to easily form equivalence classes of variables if desired. Let us denote by $G = (g_1, ..., g_l)^T$ the polynomial

vector field of the reduced system. Note that G is expressed in terms of the variables $\mathbf{y} = (y_1, ..., y_l)^T$, more precisely $G = B^T F(B\mathbf{y})$.

Corollary 6.4. *Let $p, q \in \mathbb{R}[\mathbf{x}]$. Then $p \sim_{\mathbf{\Phi}} q$ in $C_{\mathbf{\Phi}}$ if and only if $p(B\mathbf{y}) \sim_{\mathbf{\Psi}} q(B\mathbf{y})$ in $C_{\mathbf{\Psi}}$. In particular, $x_i \sim_{\mathbf{\Phi}} x_j$ in if and only if row i equals row j in B.*

Example 6.5. Let us consider again the problem of Example 3.7. Recall from Example 5.5 that the algorithm for computing invariants stops with $m = 1$ returning V_1, J_1. In particular, $V \stackrel{\triangle}{=} V_1 = \text{span}\{(1, -1, 0, 0)^T, (0, 0, 1, -1)^T\}$. It is easily checked that $W = V^{\perp} = \text{span}\{c_1, c_2\}$, where $c_1 = (1/\sqrt{2}, 1/\sqrt{2}, 0, 0)^T$ and $c_2 = (0, 0, 1/\sqrt{2}, 1/\sqrt{2})^T$ are orthonormal vectors. Hence we let the basis matrix be $B = [c_1|c_2]$. By applying (6.4), we build the minimal system in the variables $\mathbf{y} = (y_1, y_2)^T$ shown below.

$$\begin{cases} \dot{y}_1(t) &= \frac{1}{\sqrt{2}} y_1(t) y_2(t) + y_2(t) \\ \dot{y}_2(t) &= y_2(t) \\ \mathbf{y}(0) &= (0, \sqrt{2})^T . \end{cases}$$

Note that the first and second rows of B are equal, as well as the third and the fourth, which proves (again) that $x(t) = y(t)$ and $z(t) = w(t)$.

7. Approximate reduction and linearization

If one is ready to accept some degree of approximation, it is possible to build a reduced system in a simpler, linear form. To this purpose, we will illustrate a method for approximate linearization. The polynomial behaviours induced by the original and by the reduced system will in general differ by a term which is $O(t^m)$, for some prescribed m. This may be useful for analysis of the beaviour of the system around a chosen operation point t_0, which here will be conventionally fixed as $t_0 = 0$. The approximation can of course be very bad for t too far from 0 (see in the concluding section the discussion on TPWL for a strategy to possibly recover global accuracy). The method is based entirely on simple linear algebraic manipulations. Another attractive feature is that the reduced system, besides being linear, is quite small, featuring no more than m equations. We will also give conditions under which the reduction is exact.

Let $S \subseteq \mathbb{R}[\mathbf{x}]$ be a set of polynomials of bounded degree – that is, there is k such that $\deg(p) \leq k$ for each $p \in S$. For example, one might take $S = \pi[\mathbb{R}^n]$, for a certain template π with n parameters. Assume we are interested in computing/simulating the set of functions $\{p(\mathbf{x}(t)) : p \in S\} \subseteq \mathcal{A}$ and are ready to accept an approximation error around 0 of order m, for a fixed $m \geq 1$.

The basic idea is viewing $\mathcal{L}$ as a linear operator on the vector space of polynomials, and then taking its projection onto a small, suitably chosen subspace. Let $\mathcal{M} = \{\alpha_1, ..., \alpha_M\}$ be the set of distinct monomials appearing in the polynomials of $\cup_{j=0}^{m-1} \mathcal{L}^{(j)}(S)$. Let $U_{\mathcal{M}}$ be the set of polynomials that can be generated starting from $\mathcal{M}$. Clearly, $U_{\mathcal{M}} \subseteq \mathbb{R}_d[\mathbf{x}]$ for some d large enough. Moreover, when we regard polynomials as (column) vectors whose components are indexed by monomials, totally ordered in some way, $U_{\mathcal{M}}$ is isomorphic to $\mathbb{R}^M$ as a M-dimensional vector space over the field $\mathbb{R}$. Let $\text{pr}_{U_{\mathcal{M}}}$ be the orthogonal projection from $\mathbb{R}[\mathbf{x}]$ onto $U_{\mathcal{M}}$, and consider the function $\text{pr}_V \circ \mathcal{L} : U_{\mathcal{M}} \to U_{\mathcal{M}}$. We denote by L the $M \times M$

matrix representing $\mathrm{pr}_{U_\mathcal{M}} \circ \mathcal{L}$ w.r.t. the canonical basis of $U_\mathcal{M}$, that is $\mathcal{M}$. By definition, $(\mathrm{pr}_{U_\mathcal{M}} \circ \mathcal{L})_{|S} = \mathcal{L}_{|S}$. This implies that, for each $p \in S$, $\mathcal{L}(p) = Lp$, and more generally that

$$\mathcal{L}^{(j)}(p) \quad = \quad L^j p \quad (p \in S \text{ and } 0 \leq j \leq m-1). \tag{7.1}$$

Moreover, if we evaluate the given monomials at v_0 and let

$$\phi^T \quad \stackrel{\triangle}{=} \quad (\alpha_1(v_0), ..., \alpha_M(v_0))$$

then

$$(\mathcal{L}^{(j)}(p))(v_0) \quad = \quad \phi^T L^j p \quad (p \in S \text{ and } 0 \leq j \leq m-1). \tag{7.2}$$

Now consider the subspace of K_m defined as follows

$$K_m \quad \stackrel{\triangle}{=} \quad \mathrm{span}\left\{\phi, L^T\phi, ..., (L^T)^{m-1}\phi\right\} . \tag{7.3}$$

This is the order-m *Krylov space* generated by the matrix L^T and by ϕ.

Fix now any orthonormal basis of K_m, which we may represent as a $M \times l$ matrix $B = [b_1|...|b_l]$, for some $l \leq m$ (orthonormality means $B^T B = I_l$). The orthogonal projection from $U_\mathcal{M}$ onto K_m is therefore given by $\mathrm{pr}_{K_m}(v) = BB^T v$. Consider the function that, taken a vector $v \in U_\mathcal{M}$, applies L^T and then projects onto K_m

$$v \mapsto \mathrm{pr}_{K_m}(L^T v) = BB^T L^T v\,.$$

When restricted to K_m, this defines a linear morphism $K_m \to K_m$. Call A the $l \times l$ matrix representing this morphism w.r.t. the basis B. Explicitly,

$$A \quad = \quad B^T L^T B\,.$$

Lemma 7.1. *For $0 \leq j \leq m-1$, we have $(L^T)^j \phi = BA^j B^T \phi$.*

Proof. This is an easy consequence of the fact that, for any $v \in K_m$, $v = BB^T v$, and of the definition of A. □

Let us now introduce a new vector of l variables, $\mathbf{y} = (y_1, ..., y_l)^T$ and consider the following *linear* initial value problem, given by the matrix A

$$\left\{ \begin{array}{rcl} \dot{\mathbf{y}}(t) & = & A\mathbf{y}(t) \\ \mathbf{y}(0) & = & B^T\phi\,. \end{array} \right. \tag{7.4}$$

Let us denote by $\mathbf{y}(t) = (y_1(t), ..., y_l(t))^T$ the (unique) analytic solution of this system. Recall that, given $p \in \mathbb{R}[\mathbf{x}]$, we let $p(\mathbf{x}(t)) \in \mathcal{A}$ denote the function which extends $p \circ \mathbf{x}(t)$. The following theorem says that, given $p \in S$, we can reconstruct $p(\mathbf{x}(t))$, within an approximation of order m, by taking a linear combination of the $y_i(t)$ s, with coefficients given by projecting p (as a vector in $U_\mathcal{M}$) onto K_m.

Theorem 7.2. *Let $p \in S$ and let $\mathbf{y}(t)$ be the unique solution of the (reduced) problem* (7.4)*. Then $p(\mathbf{x}(t)) - p^T B\mathbf{y}(t) = O(t^m)$ as $t \to 0$.*

Proof. By definition, the derivatives of $\mathbf{y}(t)$ from the 0-th through the $(m-1)$-th, evaluated at $t = 0$, can be written as follows:

$$\begin{array}{rcl} \mathbf{y}(0) & = & B^T\phi \\ \mathbf{y}^{(1)}(0) & = & AB^T\phi \\ & \vdots & \\ \mathbf{y}^{(m-1)}(0) & = & A^{m-1}B^T\phi\,. \end{array}$$

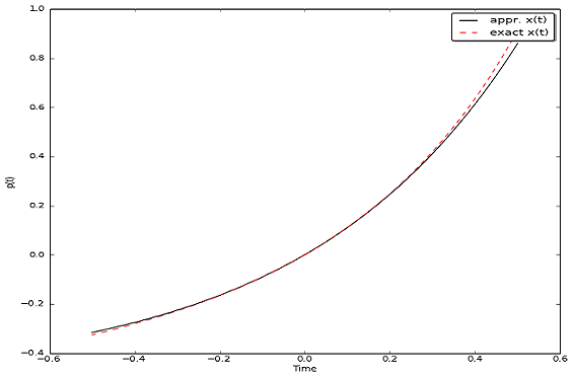


Figure 1: Plots of exact and approximate solutions.

Therefore, for $j = 0, ..., m-1$, we have

$$\begin{aligned} p^{(j)}(0) &= (\mathcal{L}^{(j)}(p))(\mathbf{x}(t))_{|t=0} \\ &= (\mathcal{L}^{(j)}(p))(v_0) \\ &= \phi^T L^j p && (7.5) \\ &= p^T (L^T)^j \phi \\ &= p^T B A^j B^T \phi && (7.6) \\ &= p^T B \mathbf{y}^{(j)}(0) && (7.7) \\ &= \frac{d^j}{dt^j} \left((p^T B)\mathbf{y}(t) \right)_{|t=0} && (7.8) \end{aligned}$$

where: (7.5) is (7.2), (7.6) follows from Lemma 7.1, (7.7) from the above expressions for the derivatives of $\mathbf{y}(t)$ at 0. This way, we have proved that the first m coefficients in the Taylor expansion around 0 of $p(\mathbf{x}(t))$ and $(p^T B)\mathbf{y}(t)$ are the same, which is the wanted statement. □

Example 7.3. Consider again the system of Example 3.7. Assume we are interested in the behaviours described by $S = \{x, y, z, w\}$. The following reduced linear system in the variables $\mathbf{y} = (y_1, y_2, y_3)^T$, which guarantees an approximation of order $m = 3$ for those behaviours, can be obtained by applying the method.

$$\begin{cases} \dot{y}_1 = 3y_1/2 + \sqrt{5}y_2/10 + \sqrt{30}y_3/30 \\ \dot{y}_2 = \sqrt{5}y_1/2 + 11y_2/10 + \sqrt{6}y_3 11/30 \\ \dot{y}_3 = \sqrt{6}y_2/5 + 4y_3/10 \\ \mathbf{y}(0) = (2, 0, 0)^T . \end{cases} \tag{7.9}$$

We do not report the full 4×3 matrix B here, but only mention that, following Theorem 7.2, the approximated version of $x(t)$ can be obtained from $\mathbf{y}(t)$ as: $\hat{x}(t) = (1, 0, 0, 0) \cdot B\mathbf{y}(t) = \sqrt{5}y_2(t)/5 - \sqrt{30}y_3(t)/10$. The plots in Figure 7 show that the approximation is quite good around $t = 0$.

We finally give a sufficient condition for exactness. This condition applies, for example, to linear systems, that is when the drifts f_i in F are linear functions. Consider the sequence of Krylov spaces $K_1, K_2, ...$ that can be built according to equation (7.3). A Krylov space

K_m is said to be *invariant* if $K_{m+1} \subseteq K_m$. Of course there always exists $m \leq M$ such that K_m is invariant; moreover the condition of invariance for K_m can be efficiently detected (see [35]).

Corollary 7.4. *Assume $\mathcal{M}$ includes all monomials in $\cup_{j\geq 0}\mathcal{L}^{(j)}(S)$. Choose m such that K_m is invariant. Then, with the notation introduced above, $p(\mathbf{x}(t)) = p^T B\mathbf{y}(t)$.*

Proof. Under the given condition, $\mathcal{L}^{(j)}(p) = Lp$ for any $j \geq 0$ and $p \in S$. Moreover, since $K_m = K_{m+1} = \cdots$, the statement in Lemma 7.1 also holds for any $j \geq m$, hence for any j. This allows one to extend the proof given in 7.2 to prove that the Taylor coefficients of $p(\mathbf{x}(t))$ and $(p^T B)\mathbf{y}(t)$ are all the same. □

Remark 7.5 (on computing the matrices A and B)**.** There exist relatively efficient and numerically stable methods to build the pair of matrices A and B needed in the construction of the reduced system. One of these methods is the *Arnoldi algorithm* [3, 35], which takes $O(Nz \cdot m)$ floating-point operations, and as much memory if a sparse storage scheme is adopted. Here Nz denotes the number of nonzero elements in the matrix L: this is $O(M^2)$ in the worst case, but it will be typically much smaller, as polynomials arising in applications tend to be sparse. Moreover, the method need not a fully stored matrix L, but only a handle to the matrix-vector multiplication function $u \mapsto L^T u$. Using appropriate data structures, this method can be implemented according to an 'on-the-fly' strategy, where the terms are unfolded as needed.

8. Examples

Although the focus of the present paper is mostly on theory, it is instructive to put a proof-of-concept implementation[7] of our algorithms at work on a few simple examples taken from the literature. We illustrate below two cases, a linear and a nonlinear system.

8.1. **Example 1: linearity and weighted automata.** The purpose of this example is to argue that, when the transition structure induced by the Lie-derivatives is considered, $\mathcal{L}$-bisimilarity is fundamentally a linear-time semantics. We first introduce *weighted automata*, a more compact[8] way of representing the transition structure of the coalgebra $C_{\mathbf{\Phi}}$.

A (finite- or infinite-state) weighted automaton is like an ordinary automaton, save that both states and transitions are equipped with *weights* from $\mathbb{R}$. Given F and v_0, we can build a weighted automaton with monomials as states, weighted transitions given by the rule $\alpha \xrightarrow{\lambda} \beta$ iff $\mathcal{L}_F(\alpha) = \lambda\beta + q$ for some polynomial q not comprising β as a monomial, and real $\lambda \neq 0$, and where each state α is assigned weight $\alpha(v_0)$. As an example, consider the weighted automaton in Figure 2, where the state weights (not displayed) are 1 for x_{10}, and 0 for any other state. This automaton is generated – and in fact codes up – a system of ODE's with ten variables, where $\dot{x}_1 = x_2$, $\dot{x}_2 = (2/3)x_3 + (1/3)x_4$ etc., with the initial condition as specified by the state weights ($x_1(0) = 0$ etc.).

A *run* in a weighted automaton is a path in the graph from a state to a state. The run's weight is the product of all involved transition weights *and* the last state's weight.

[7] Python code available at `http://local.disia.unifi.it/boreale/papers/DoubleChain.py`. Reported execution times relative to the pypy interpreter under Windows 8 on a core i5 machine.

[8] At least for linear vector fields, the resulting weighted automaton is finite.

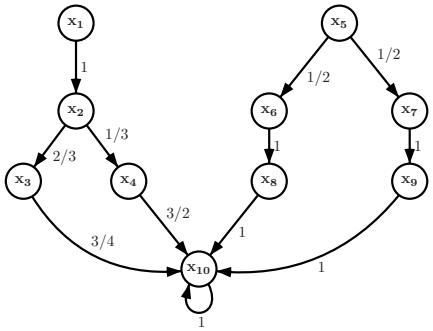
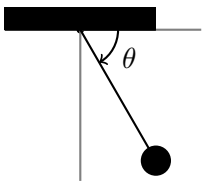


Figure 2: A weighted automaton.

In the standard linear-time semantics of weighted automata, each state α is assigned a function $\sigma_\alpha : \mathbb{N} \to \mathbb{R}$ (a *stream*, in the terminology of Rutten [34]) such that $\sigma_\alpha(i)$ is obtained by summing up all the weights of the runs involving i transitions starting from s; e.g. $\sigma_{x_1}(3) = 1 \times \frac{2}{3} \times \frac{3}{4} + 1 \times \frac{1}{3} \times \frac{3}{2} = 1$. Standard results in coalgebra (or a simple direct proof) ensure that this semantics is in agreement with bisimulation, in the sense that for any pair of states (in our case, monomials) α and β, $\alpha \sim_{\mathbf{\Phi}} \beta$ if and only $\sigma_\alpha = \sigma_\beta$; we refer the interested reader to [34, 6] for details on this construction and result.

When applied to our example, these results imply for instance that $x_1(t) = x_5(t)$. In fact, when invoked with this system and $\pi = \sum_{i=1}^{10} a_i x_i$ as inputs, the double chain algorithm terminates at $m = 2$ (in about 0.3 s; this being a linear system, Gröbner bases are never actually needed), returning $\pi[V_2] = a_1(x_6 - x_7) + a_2(x_8 - x_9) + a_3(x_6 - x_2) + a_4(x_5 - x_1) + a_5(\frac{3}{2}x_8 - x_4) + a_6(\frac{3}{4}x_8 - x_3)$. This implies the expected equality $x_1 = x_5$ (let $a_4 = 1$ and $a_i = 0$ for $i \neq 4$ in the returned template), as well as other equalities, such as $x_2 = x_6 = x_7$ and $x_8 = x_9$. All in all, V_2 being a 6-dimensional space, we will have a 4-dimensional $W = V_2^\perp$, that is a minimal system with 4 equations, a 60% reduction.

8.2. **Example 2: nonlinear conservation laws.** The law of the simple pendulum is $\frac{d^2}{dt^2}\theta = \frac{g}{\ell}\cos\theta$, where θ is the angle from the roof to the rod measured clockwise, ℓ is the length of the rod and g is gravity acceleration (see picture on the right). If we assume the initial condition $\theta(0) = 0$, this can be translated into the polynomial initial value problem below, where $\mathbf{x} = (\theta, \omega, x, y)^T$. The meaning of the variables is $\omega = \dot\theta$, $x = \cos\theta$ and $y = \sin\theta$. We assume for simplicity $\ell = 1$ and $g = 9$.



$$\begin{cases} \dot\theta &= \omega \\ \dot\omega &= \frac{g}{\ell}x \\ \dot x &= -y\omega \\ \dot y &= x\omega \\ \mathbf{x}(0) &= (0,0,\ell,0)^T . \end{cases}$$

For this system, the double chain algorithm reports that there is no nontrivial linear conservation law (after $m = 6$ iterations and about 0.3 s). We then ask the algorithm to find all the conservation laws of order two, that is we use the template (α ranges over monomials) $\pi = \sum_{\alpha_i\,:\,\deg(\alpha_i)\leq 2} a_i\alpha_i$ as input. The algorithm terminates after $m = 16$ iterations (in about 7 s). The invariant J_{16} contains all the wanted conservation laws. The returned Gröbner basis for it is $G = \{x^2 + y^2 - 1, \omega^2 - 18y\}$. The first term here just expresses the

trigonometric identity $(\cos\theta)^2 + (\sin\theta)^2 = 1$. Recalling that the (tangential) speed of the bob is $v = \ell\dot{\theta} = \ell\omega$, and that its vertical distance from the roof is $h = \ell\sin\theta = \ell y$, we see that the second term, considering our numerical values for ℓ, g, is equivalent to the equation $\frac{1}{2}v^2 = gh$, which, when multiplied by the mass m of the bob, yields the law of conservation of energy $\frac{1}{2}mv^2 = mgh$ (acquired kinetic energy = lost potential energy).

9. Conclusion, further and related work

We have presented a framework for automatic reasoning and reduction in systems of polynomial ODE's. In particular, we offer algorithms to: (1) compute the most general set of identities valid in the system that fit a user-specified template; and, (2) build a minimal system equivalent to the original one. These algorithms are based on a mix of simple algebraic and coalgebraic techniques.

9.1. **Directions for further work.** Scalability of our approach is an issue, as, already for simple systems, the Gröbner basis construction involved in the double chain algorithm can be computationally quite demanding. Further experimentation, relying on a well-engineered implementation of the method, and considering sizeable case studies, is called for in order to assess this aspect. One would also like to extend the present approach so as to deal with *regions* of possible initial values, rather than fixing one such value. This is important, e.g., in the treatment of hybrid systems (see below). After the short version of the present paper [9] appeared, some preliminary progress towards this goal has been made, see [10]. Concerning minimization, we note that the reduced system may not preserve the structure of the original one, e.g. as to the meaning of variables: this may be problematic in certain application domains, such as system biology. In the future, we intend to investigate this issue. Approximate reductions in the sense of System Theory [1] are also worth investigating. One problem of the approach described in Section 7 is that the reduced system depends on a fixed initial condition v_0. Obtaining bounds on the approximation error is another aspect that deserves further investigation. Some preliminary progress on these issues is reported in [11].

9.2. **Related work.** Bisimulations for weighted automata are related to our approach, because, as argued in subsection 8.1, Lie-derivation can be naturally represented by such an automaton. Algorithms for computing largest bisimulations on *finite* weighted automata have been studied by Boreale et al. [7, 6]. A crucial ingredient in these algorithms is the representation of bisimulations as finite-dimensional vector spaces. Approximate versions of this technique have also been recently considered in relation to Markov chains [8]. As discussed in Remark 5.7, in the case of linear systems, the algorithm in the present paper reduces to that of [7, 6]. Algebraically, moving from linear to polynomial systems corresponds to moving from vector spaces to ideals, hence from linear bases to Gröbner bases. From the point of view automata, this step leads to considering infinite weighted automata. In this respect, the present work may be also be related to the automata-theoretic treatment of linear ODE's by Fliess and Reutenauer [21].

Although there exists a rich literature dealing with linear aggregation of systems of ODE's (e.g. [1, 25, 40, 27]), we are not aware of fully automated approaches to minimization (Theorem 6.3), with the notable exception of a series of recent works by Cardelli and

collaborators [14, 15, 16]. Mostly related to ours is [14]. There, for an extension of the polynomial ODE format called IDOL, the authors introduce two flavours of *differential equivalence*, called *Forward* (FDE) and *Backward* (BDE). They provide a symbolic, SMT-based partition refining algorithms to compute the largest equivalence of each type. FDE groups variables in such a way that the corresponding quotient system recovers the *sum* of the original solutions in each class, whatever the initial condition. However, precise information on the individual original solutions cannot in general be recovered from the reduced system. In BDE, variables grouped together are guaranteed to have the same solution. Therefore the quotient system permits in this case to fully recover the original solutions. As such, BDE can be compared directly to our $\mathcal{L}$-bisimulation.

An important difference is that BDE may tell apart variables that have the same solution. As already seen, this is not the case with $\mathcal{L}$-bisimilarity, which is correct and complete. An important consequence of this difference is that the quotient system produced by BDE is not minimal, whereas that produced by $\mathcal{L}$-bisimulation is, in a precise sense. In concrete cases, this may imply a significant size difference. For example, in the linear system of Subsection 8.1, BDE finds only two equalities[9], leading to a quotient system of eight states; on the other hand, the minimal system produced by $\mathcal{L}$-bisimulation has four states. For what concerns reasoning, we note that, being based on partitions of variables, BDE cannot express relations involving polynomial, or even linear, combinations of variables. Finally, the approach of [14] and ours rely on two quite different algorithmic decision techniques, SMT and Gröbner bases, both of which have exponential worst-case complexity. As shown by the experiments reported in [14], in practice BDE and FDE have proven quite effective at system reduction. At the moment, we lack similar experimental evidence for $\mathcal{L}$-bisimilarity. We also note that, limited to the case of polynomials of degree two, a polynomial algorithm for BDE exists [15].

Linear aggregation and lumping of (polynomial) systems of ODE's are well known in the literature, se e.g. [1, 27, 25, 40] and references therein. However, as pointed out by Cardelli et al. [14], no general algorithms for computing the largest equivalence, hence the minimal *exact* reduction (in the sense of our Theorem 6.3) was known.

The seminal paper of Sankaranarayanan, Sipma and Manna [37] introduced polynomial ideals to find invariants of hybrid systems. Indeed, the study of the safety of hybrid systems can be shown to reduce constructively to the problem of generating invariants for their differential equations [30]. The results in [37] have been subsequently refined and simplified by Sankaranarayanan using *pseudoideals* [38], which enable the discovery of polynomial invariants of a special form. Other authors have adapted this approach to the case of imperative programs, see e.g. [12, 26, 32] and references therein. Reduction and minimization seem to be not a concern in this field.

Platzer has introduced *differential dynamic logic* to reason on hybrid systems [29]. The rules of this logic implement a fundamentally inductive, rather than coinductive, proof method. Mostly related to ours is Ghorbal and Platzer's recent work on polynomial invariants [23]. One one hand, they characterize algebraically invariant regions of vector fields – as opposed to initial value problems, as we do. On the other hand, they offer sufficient conditions under which the trajectories induced by specific initial values satisfy all instances of a polynomial template (cf. [23, Prop.3]). The latter result compares with ours, but the resulting method appears to be not (relatively) complete in the sense of our double chain algorithm. Moreover, the computational prerequisites of [23] (symbolic linear programming,

[9]Which are $x_6 = x_7$ and $x_8 = x_9$, as checked with the Erode tool by the same authors [17].

exponential size matrices, symbolic root extraction) are very different from ours, and much more demanding. Again, minimization is not addressed.

Ideas from Algebraic Geometry have been fruitfully applied also in Program Analysis. Relevant to our work is Müller-Olm and Seidl's [26], where an algorithm to compute all polynomial invariants up to a given degree of an imperative program is provided. Similarly to what we do, they reduce the core problem to a linear algebraic one. However, being the setting in [26] discrete rather than continuous, the techniques employed there are otherwise quite different, mainly because: (a) the construction of the ideal chain is driven by the program's operational semantics, rather than by Lie derivatives; (b) the found polynomial invariants must be valid under *all* initial program states, not just under the user specified one. If transferred to a continuous setting, condition (b) would lead in most cases to trivial invariants.

In nonlinear Control Theory, there is a huge amount of literature on *Model Order Reduction* (MOR), that aims at reducing the size of a given system, while preserving some properties of interest, such as stability and passivity. A well established approach relies on building truncated Taylor expansions of the given sysytems [24, 28], repeated at various points along a trajectory of interest, to keep the approximation error globally small: a technique known as *trajectory piece-wise linear* (TPWL) MOR, see e.g. [31]. One wonders whether our approximate linearization technique of Section 7 might conveniently serve as a building block of this strategy.

The present paper is the extended and revised version of [9]. W.r.t. [9], here we include complete proofs, the discussion of up-to techniques in Section 3, the approximate linearization technique of Section 7 and an extended and updated discussion of further and related work in the present section.

Acknowledgments

The author has benefited from stimulating discussions with Mirco Tribastone. Two anonymous LMCS reviewers have provided valuable comments that have helped to improve the presentation.

References

[1] A.C. Antoulas. *Approximation of Large-scale Dynamical Systems*. SIAM, 2005.
[2] V.I. Arnold. *Ordinary Differential Equations*. The MIT Press, ISBN 0-262-51018-9, 1978.
[3] W. E. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics*, vol. 9, pp. 17-29, 1951.
[4] M. Bernardo. A survey of Markovian behavioral equivalences. In *Formal Methods for Performance Evaluation*, vol. 4486 of LNCS, pages 180-219. Springer, 2007.
[5] M. L. Blinov, J. R. Faeder, B. Goldstein, and W. S. Hlavacek. BioNet-Gen: software for rule-based modeling of signal transduction based on the interactions of molecular domains. *Bioinformatics*, 20(17): 3289-3291, 2004.
[6] F. Bonchi, M.M. Bonsangue, M. Boreale, J.J.M.M. Rutten, and A. Silva. A coalgebraic perspective on linear weighted automata. *Inf. Comput.* 211: 77-105, 2012.
[7] M. Boreale. Weighted Bisimulation in Linear Algebraic Form. *Proc. of CONCUR 2009*, LNCS 5710, pp. 163-177, Springer, 2009.
[8] M. Boreale. Analysis of Probabilistic Systems via Generating Functions and Padé Approximation. *ICALP 2015* (2): 82-94, LNCS 9135, Springer, 2015. Extended version available as DiSIA working paper 2016/10, `http://local.disia.unifi.it/wp_disia/2016/wp_disia_2016_10.pdf`.

[9] M. Boreale. Algebra, coalgebra, and minimization in polynomial differential equations. In *Proc. of FoSSACS 2017*:71–87, LNCS 10203, Springer, 2017. Conference version of the present paper.

[10] M. Boreale. Complete algorithms for algebraic strongest postconditions and weakest preconditions in polynomial ODE's. In *Proc. of SOFSEM'18*, LNCS 10706: 442–455, Springer. Full version available as `http://arxiv.org/abs/1708.05377`, 2017.

[11] M. Boreale. Algorithms for exact and approximate linear abstractions of polynomial continuous systems. *HSCC 2018*: 207–216, ACM, 2018.

[12] D. Cachera, Th. Jensen, A. Jobi, and F. Kirchner. Inference of Polynomial Invariants for Imperative Programs: A Farewell to Gröbner Bases. *SAS 2012*, LNCS 7460: 58-74, Springer, 2012.

[13] L. Cardelli. On process rate semantics. *Theoretical Computer Science*, 391(3):190-215, 2008.

[14] L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin. Symbolic Computation of Differential Equivalences, *POPL 2016*, ACM, 2016.

[15] L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin. Efficient Syntax-driven Lumping of Differential Equations, *TACAS 2016*:93-111, LNCS 9636, Springer, 2016.

[16] L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin. Comparing Chemical Reaction Networks: A Categorical and Algorithmic Perspective, *LICS 2016*, IEEE, 2016.

[17] L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin. ERODE: Evaluation and Reduction of Ordinary Differential Equations. Available from `http://sysma.imtlucca.it/tools/erode/`.

[18] F. Ciocchetta and J. Hillston. Bio-PEPA:A framework for the modelling and analysis of biological systems. *Theoretical Computer Science*, 410 (33-34):3065-3084, 2009

[19] M. Colón. Polynomial approximations of the relational semantics of imperative programs. *Science of Computer Programming* 64: 76-9, 2007.

[20] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Undergraduate Texts in Mathematics, Springer, 2007.

[21] M. Fliess, and C. Reutenauer. Theorie de Picard-Vessiot des Systèmes Reguliers. *Colloque Nat. CNRS-RCP567, Belle-ile sept. 1982*, in *Outils et Modèles Mathématiques pour l'Automatique l'Analyse des systèmes et le tratement du signal*. CNRS, 1983.

[22] H.P. Gumm, Tobias Schröder. Covarieties and complete covarieties. *Theoretical Computer Science* 260(1):71-86, Elsevier, 2001.

[23] K. Ghorbal, A. Platzer. Characterizing Algebraic Invariants by Differential Radical Invariants. *TACAS 2014*: 279-294, 2014. Extended version available from `http://reports-archive.adm.cs.cmu.edu/anon/2013/CMU-CS-13-129.pdf`.

[24] P. Li and L. Pileggi. Compact reduced-order modeling of weakly nonlinear analog and RF circuits. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 24, no. 2, pp. 184-203, 2005.

[25] G. Li, H. Rabitz, and J. Tóth. A general analysis of exact nonlinear lumping in chemical kinetics. *Chemical Engineering Science* 49 (3), 343-361, 1994.

[26] M. Müller-Olm and H. Seidl. Computing polynomial program invariants. *Information Processing Letters* 91(5), 233-244, 2004.

[27] M.S. Okino and M.L. Mavrovouniotis. Simplification of mathematical models of chemical reaction systems. *Chemical Reviews*, 2(98):391-408, 1998.

[28] J.R. Phillips. Projection-based approaches for model reduction of weakly nonlinear time-varying systems. *IEEE Trans. Comput.-Aided Des.*, vol. 22, no. 2, pp. 171-187, 2003.

[29] A. Platzer. Differential dynamic logic for hybrid systems. *J. Autom. Reasoning* 41(2), 143-189, 2008.

[30] A. Platzer. Logics of dynamical systems. In *LICS 2012*: 13-24, IEEE, 2012.

[31] M. Rewienski and J. White. A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices. *IEEE Trans. Comput.-Aided Des.*, vol. 22, no. 2, pp. 155-170, 2003.

[32] E. Rodríguez-Carbonell and D. Kapur. Generating all polynomial invariants in simple loops. *Journal of Symbolic Computation* 42(4), 443-476, 2007.

[33] J. Roychowdhury. Reduced-order modelling of time-varying systems. *IEEE Trans. Circuits Syst.-II: Analog Digital Signal Process.*, vol. 46, no. 10, pp. 1273-1288, 1999.

[34] J.J.M.M. Rutten. Behavioural differential equations: a coinductive calculus of streams, automata, and power series. *Theoretical Computer Science*, 308(1–3): 1–53, 2003.

[35] Y. Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.

[36] D. Sangiorgi. Beyond Bisimulation: The "up-to" Techniques. *FMCO 2005*: 161-171, 2005.

[37] S. Sankaranarayanan, H. Sipma, and Z. Manna. Non-linear loop invariant generation using Gröbner bases. *POPL 2004*.
[38] S. Sankaranarayanan. Automatic invariant generation for hybrid systems using ideal fixed points. *HSCC 2010*: 221-230, ACM, 2010.
[39] A. Tiwari. Approximate reachability for linear systems. *HSCC 2003*: 514-525, ACM, 2003.
[40] J. Tóth, G. Li, H. Rabitz, and A. S. Tomlin. The effect of lumping and expanding on kinetic differential equations. *SIAM Journal on Applied Mathematics*, 57(6):1531-1556, 1997.
[41] M. Tribastone, S. Gilmore, and J. Hillston. Scalable differential analysis of process algebra models. *IEEE Trans. Software Eng.*, 38(1):205-219, 2012.
[42] E. O. Voit. Biochemical systems theory: A review. *ISRN Biomathematics*, 2013:53, 2013.

Appendix A. Proofs and additional technical material

A.1. **Proofs.**

Proof of Theorem 4.5. We check each of the equivalences (4.2–4.5) in turn.

- If $p \sim_{\mathbf{\Phi}} q$ then $p - q \in \ker(\sim_{\mathbf{\Phi}})$ by definition of ker. Conversely, assume $p - q \in \ker(\sim_{\mathbf{\Phi}}) \subseteq \left\langle\ \ker(\sim_{\mathbf{\Phi}})\ \right\rangle$. Since, by first part of Lemma 4.3 with $R = \sim_{\mathbf{\Phi}}$, the last set is an invariant, by the second part of the same lemma we obtain the wanted $p \sim_{\mathbf{\Phi}} q$, hence (4.2).
- Since $p \sim_{\mathbf{\Phi}} q$ implies $(p-q)(t) = 0$, and $r(t) = 0$ implies $r \sim_{\mathbf{\Phi}} 0$ (in both cases by Theorem 3.2), from the definition of $\ker(\sim_{\mathbf{\Phi}})$ the second equality (4.3) immediately follows.
- A polynomial behaviour $p(\mathbf{x}(t))$ is identically 0 in $\mathcal{A}$ if and only if all its derivatives $\frac{d}{dt^j} p(\mathbf{x}(t))$, $j \geq 0$, vanish at 0, and this, via (2.7) and (2.5), yields (4.4).
- Note that if $p \in I$, with I an invariant, then $p^{(j)}(v_0) = 0$ for each $j \geq 0$ (easily shown by induction on j). Conversely, if $p^{(j)}(v_0) = 0$ for each j, then $J = \left\langle\ \{p, p^{(1)}, p^{(2)}, ...\}\ \right\rangle$ is an invariant and contains p. To check invariance of J, consider a generic $q \in J$, $q = \sum_i h_i p^{(j_i)}$: it is immediate that $q(v_0) = 0$ and that $\mathcal{L}(q) = \sum_i \mathcal{L}(h_i) p^{(j_i)} + \sum_i h_i p^{(j_i+1)} \in J$. This way we have also proven the last equation (4.5). □

Proof of Lemma 5.2. We proceed by induction on j. The base case $j = 1$ follows from the definition of m. Assuming by induction hypothesis that $V_m = \cdots = V_{m+j}$ and that $J_m = \cdots = J_{m+j}$, we prove now that $V_m = V_{m+j+1}$ and that $J_m = J_{m+1+1}$. The key to the proof is the following fact

$$\pi^{(m+j+1)}[v] \quad \in \quad J_m \quad \text{for each } v \in V_m\,. \tag{A.1}$$

From this fact the thesis will follow, indeed:

(1) $V_m = V_{m+j+1}$. To see this, observe that for each $v \in V_{m+j} = V_m$ (the equality here follows from the induction hypothesis), it follows from (A.1) that $\pi^{(m+j+1)}[v]$ can be written as a finite sum of the form $\sum_l h_l \cdot \pi^{(j_l)}[u_l]$, with $0 \leq j_l \leq m$ and $u_l \in V_m$. As a consequence, $\pi^{(m+j+1)}[v](v_0) = 0$, which shows that $v \in V_{m+j+1}$. This proves that $V_{m+j+1} \supseteq V_{m+j} = V_m$; the reverse inclusion is obvious;
(2) $J_m = J_{m+j+1}$. As a consequence of $V_{m+j+1} = V_{m+j} (= V_m)$ (the previous point), we can write

$$\begin{aligned} J_{m+j+1} &= \left\langle\ \cup_{i=0}^{m+j} \pi^{(i)}[V_{m+j}] \cup \pi^{(m+j+1)}[V_{m+j}]\ \right\rangle \\ &= \left\langle\ J_{m+j} \cup \pi^{(m+j+1)}[V_{m+j}]\ \right\rangle \\ &= \left\langle\ J_m \cup \pi^{(m+j+1)}[V_m]\ \right\rangle \end{aligned}$$

where the last step follows by induction hypothesis. From (A.1), we have that $\pi^{(m+j+1)}[V_m] \subseteq J_m$, which implies the thesis for this case, as $\left\langle\ J_m\ \right\rangle = J_m$.

We prove now (A.1). Fix any $v \in V_m$. First, note that $\pi^{(m+j+1)}[v] = \mathcal{L}(\pi^{(m+j)}[v])$ (here we are using (5.1)). As by induction hypothesis $\pi^{(m+j)}[V_m] = \pi^{(m+j)}[V_{m+j}] \subseteq J_{m+j} = J_m$, we have that $\pi^{(m+j)}[v]$ can be written as a finite sum $\sum_l h_l \cdot \pi^{(j_l)}[u_l]$, with $0 \leq j_l \leq m$ and $u_l \in V_m$. Applying the rules of Lie derivatives (2.2), (2.3), we find that $\pi^{(m+j+1)}[v] = \mathcal{L}(\pi^{(m+j)}[v])$ equals

$$\sum_l \left(h_l \cdot \pi^{(j_l+1)}[u_l] + \mathcal{L}(h_l) \cdot \pi^{(j_l)}[u_l] \right) .$$

Now, for each u_l, $u_l \in V_m = V_{m+1}$, each term $\pi^{(j_l+1)}[u_l]$, with $0 \leq j_l + 1 \leq m+1$, is by definition in $J_{m+1} = J_m$. This shows that $\pi^{(m+j+1)}[v] \in J_m$, as required. □

Proof of Corollary 6.4. Concerning the first part, let us denote by $\mathcal{Z}_{\mathbf{\Psi}}$ the largest invariant induced by $\mathbf{\Psi}$ in the polynomial ring $\mathbb{R}[\mathbf{y}]$, according to Theorem 4.5. We have: $p \sim_{\mathbf{\Phi}} q$ in $\mathbb{R}[\mathbf{x}]$ if and only if $(p-q) \in \mathcal{Z}_{\mathbf{\Phi}}$ (Theorem 4.5(4.3)) if and only if $(p-q) \circ \mathbf{x}(t) = 0$ (Theorem 4.5(4.2)) if and only if $(p-q) \circ B\mathbf{y}(t) = 0$ (Theorem 6.2) if and only if $(p(B\mathbf{y}) - q(B\mathbf{y}))\,(t) = 0$ if and only if $(p(B\mathbf{y}) - q(B\mathbf{y})) \in \mathcal{Z}_{\mathbf{\Psi}}$ (Theorem 4.5(4.2)) if and only if $p(B\mathbf{y}) \sim_{\mathbf{\Psi}} q(B\mathbf{y})$ in $\mathbb{R}[\mathbf{y}]$ (Theorem 4.5(4.3)).

Concerning the second part, denoting by e_i the i-th canonical vector in $\mathbb{R}^N$, note that: $x_i \sim_{\mathbf{\Phi}} x_j$ if and only if $x_i - x_j \in \mathcal{Z}_{\mathbf{\Phi}}$ if and only if $e_i - e_j \in V_m$ if and only if $e_i - e_j \perp W$ if and only if $B^T(e_i - e_j) = 0$, from which the thesis follows for this case. □

A.2. **Pseudoideals.** We briefly discuss a sufficient condition for establishing the condition (5.7), that is $J_{i+1} = J_i$, which does not involve Gröbner bases, but only linear algebraic computations, and can therefore lead to a gain in efficiency. We ill make use of a bit of new notation. For a set of polynomials S and an integer $k \geq 0$, denote by $\left\langle\ S\ \right\rangle_k$ the subset of $\left\langle\ S\ \right\rangle$ generated from S by only using multiplier polynomials h_j of degree $\leq k$ (cf. equation (4.1)); this is a *pseudo ideal of degree k*, in the terminology of Colón [19]. We can choose $k \geq 0$ and replace (5.7) by the following stronger condition

$$\pi^{(i+1)}[B_i] \quad \subseteq \quad \left\langle\ \cup_{j=0}^{i} \pi^{(j)}[B_i]\ \right\rangle_k . \tag{A.2}$$

If (A.2) is true then of course also (5.7) is true, while the converse is not valid in general. Condition (A.2) can be checked by linear algebraic techniques, which do not involve Gröbner bases computations. Indeed, a pseudo ideal $\left\langle\ S\ \right\rangle_k$ has the structure of a vector space over $\mathbb{R}$ of dimension $|S| \cdot M$, where M is the number of distinct monomials of degree $\leq k$. However, the resulting algorithm is not guaranteed to terminate.

A.3. **Building an orthonormal basis of $V^\perp$.** We use here to the terminology of Section 6. Let us first work out a convenient characterization of the space $W = V^\perp$. Consider the successive Lie derivatives of the vector $\mathbf{x} = (x_1, ..., x_N)^T$, taken componentwise, that is the vectors of polynomials $\mathbf{x}^{(j)} = (x_1^{(j)}, ..., x_N^{(j)})^T$, for $j = 0, 1, ...$ Once evaluated at v_0, these become vectors in $\mathbb{R}^N$. We claim that

$$W \quad = \quad \text{span} \left\{ \mathbf{x}(v_0), \mathbf{x}^{(1)}(v_0), ..., \mathbf{x}^{(m)}(v_0) \right\} . \tag{A.3}$$

Indeed, we have, by definition of V (here $v = (\lambda_1, ..., \lambda_N)^T$ denotes a generic vector in $\mathbb{R}^N$)

$$\begin{aligned} V &= V_m \\ &= \left\{ v \,:\, \sum_{i=1}^{N} \lambda_i x_i^{(j)}(v_0) = 0 \text{ for } j = 0, ..., m \right\} \\ &= \left\{ v \,:\, \langle v, \mathbf{x}^{(j)}(v_0) \rangle = 0 \text{ for } j = 0, ..., m \right\} \\ &= \left\{ \mathbf{x}(v_0), \mathbf{x}^{(1)}(v_0), ..., \mathbf{x}^{(m)}(v_0) \right\}^{\perp} \end{aligned}$$

which is equivalent to (A.3). Note that $l = \dim(W) \leq m + 1, N$ (and typically one will have $l \ll N$). Therefore, a way to build an orthonormal basis B for W is just to apply the Gram-Schmidt orthonormalization process to the set of vectors on the right-hand side of (A.3). This process can in fact carried out incrementally, as the vectors of Lie derivatives $\mathbf{x}^{(j)}$ are computed.