Functional sets with typed symbols: Mixed zonotopes and Polynotopes for hybrid nonlinear reachability and filtering

Christophe Combastel

Functional sets with typed symbols : Mixed zonotopes and Polynotopes for hybrid nonlinear reachability and filtering

C. Combastel [a]

[a] *Univ. Bordeaux, CNRS, IMS, UMR 5218, 33405 Talence, France*

Abstract

Verification and synthesis of Cyber-Physical Systems (CPS) are challenging and still raise numerous issues so far. In this paper, based on a new concept of mixed sets defined as function images of symbol type domains, a compositional approach combining eager and lazy evaluations is proposed. Syntax and semantics are explicitly distinguished. Both continuous (interval) and discrete (signed, boolean) symbol types are used to model dependencies through linear and polynomial functions, so leading to mixed zonotopic and polynotopic sets. Polynotopes extend sparse polynomial zonotopes with typed symbols. Polynotopes can both propagate a mixed encoding of intervals and describe the behavior of logic gates. A functional completeness result is given, as well as an inclusion method for elementary nonlinear and switching functions. A Polynotopic Kalman Filter (PKF) is then proposed as a hybrid nonlinear extension of Zonotopic Kalman Filters (ZKF). Bridges with a stochastic uncertainty paradigm are briefly outlined. Finally, several discrete, continuous and hybrid numerical examples including comparisons illustrate the effectiveness of the theoretical results.

Key words: Functional sets; Polynomial dependencies; Mixed encoding; Logic; Hybrid dynamic systems; Reachability; Robust state estimation; Kalman filters; Zonotopes; Polynotopes;

1 Introduction

Uncertainty management undoubtedly remains a great challenge when designing, observing, controlling and verifying systems with stringent safety, reliability and accuracy requirements. Common industrial practice still makes intensive use of Monte-Carlo simulations and design of experiments to check for robustness, perform sensitivity analysis and optimize tuning. Meanwhile, given some model of the available knowledge, which is by essence subject to uncertainties, formal methods provide verification and synthesis tools likely to ensure a full coverage wrt to the range of specified behaviors, including off-nominal and worst cases. Dealing with complex dynamics such as nonlinear and hybrid ones remains challenging and the achieved trade-off between computation time and accuracy heavily depends on underlying set representations.
However, a direct use of set-membership techniques (e.g. intervals, ellipsoids, etc) is often subject to the so-called dependency problem and/or the wrapping effect. The former comes from the loss of variable multi-occurrences when overloading basic operators. The latter results from the necessary approximate (usually outer) description of true solution sets. This has motivated the use of affine arithmetic [11,37] and other set representations of intermediate complexity between intervals [18,35,10] or ellipsoids [24] and polytopes [16] or level sets [30], like zonotopes [23,4,14], a class of convex and centrally symmetric polytopic sets defined as the affine image of a unit hypercube. Zonotopes have been used to address the reachability of linear [34,13,8], nonlinear [2,3,15] and hybrid e.g. [26] systems, as well as state bounding observation [4,1,5,25], possibly with links to a stochastic paradigm [6,7], or in a distributed context [31,9]. With zonotopes, affine function transforms correspond to implicit set operations and the evaluation of bounds can be delayed (lazy evaluation [39]), to the benefit of a better management of dependencies.
Moreover, an analogy can be noticed between such affine function transforms and the manipulation of symbolic expressions at a syntactic level (e.g. $x - x$ simplified as 0 before substituting the unit interval $[-1, +1]$ for x in $x - x$). In addition, the important distinction between syntax (e.g. formal transformation rules) and semantics (e.g. the interpretation/evaluation of some expression)

Email address: `christophe.combastel@u-bordeaux.fr` (C. Combastel).

is easily lost when directly operating sets. By extending ideas originating from affine arithmetic [11] and further developed, e.g., for the static analysis of programs by abstract interpretation [14], symbolic zonotopes and USP (Unique Symbol Provider) in [9] showed the relevance of these concepts in a context of distributed state estimation. To summarize, a clear distinction between syntax and semantics is a key point to struggle against the dependency problem in set-membership computations.

When addressing core problems like reachability, state estimation, identification, invariant sets [40] and fault diagnosis [41,38,33], non-convex and possibly non-connected sets are often required when dealing with nonlinear and/or hybrid systems. Taylor models [27] are an alternative to represent non-convex sets. Dealing with non convex and/or non-connected sets is also possible through pavings [18] or level sets [30] but costly since related algorithms respectively rely on bisections or grids, both yielding an exponential complexity. Using properties like monotony/cooperativity [36] often impose restrictions either on the class of dynamics that can readily be handled, or on accuracy due to the lack of richer internal set descriptions. In the hybrid case, the crossing of guards can generate many pieces of flows. Bissection/branching may be used, but the benefit of fast methods is then often lost due to the complexity induced by the propagation of a large number of (possibly smaller) instances which unduly become fully independent after branching/bissection.
Thus, two complementary directions can be considered and *combined*: finding more versatile set representations possibly *i*) *non convex* while preserving scalability, and *ii*) *non connected* to propagate unions/bundles of a possibly large number of (implicit) sets characterizing distinct/discrete configurations/modes, without full bissections/branching, that is, while sharing and keeping trace of the common features between all these sets. The combination of *i*) and *ii*) pleads in favor of searching for some kind of unified representation to encode and operate *mixed sets* (i.e. hybrid sets).

Though zonotopic sets catch some linear dependencies, their convex, connected, and centrally symmetric nature still impose restrictions to address the reachability of nonlinear and hybrid (i.e. mixed continuous/discrete) dynamic systems. To overcome these restrictions, Taylor models [27], polynomial zonotopes [2] and sparse polynomial zonotopes (spz) [22] rely on sets defined as polynomial images rather than affine/linear ones. spz can thus efficiently store and operate a large class of non convex continuous sets. However, spz do not natively handle the case of discrete or mixed sets, which motivates the distinct features introduced with the polynotopes proposed in this paper. Moreover, constraints can be also introduced in set representations as with constrained zonotopes [38] and, recently, constrained polynomial zonotopes [21]. Note that the evaluation of bounds under constraints, even if delayed, may be costly and involve iterative algorithms (e.g. linear programming with constrained zonotopes). Polynotopes will thus introduce typed symbols whose management involves specific polynomial constraints that can be efficiently handled.

Contributions. This paper introduces a new concept of jointly mathematical and computational objects called polynotopes allowing to define and operate functional sets with typed symbols. By focusing on one continuous (interval) and two discrete (signed, boolean) symbol types, it is shown how the resulting non convex, non centrally symmetric and non connected mixed sets extending zonotopes (including polynomial ones) can be used to implement advanced hybrid nonlinear reachability and filtering algorithms without relying on costly bissections. Syntax and semantics are explicitly distinguished while combining both a strict/eager evaluation of polynotope objects (mainly addressing the dependency problem) and a lazy/delayed evaluation of bounding sets (mainly addressing the wrapping effect). Mixed encoding of intervals and dependency preserving inclusion methods for elementary nonlinear and switching functions are proposed. Polynomial representation of logic functions defined on $\{-1,+1\}$ (signed logic) or $\{0,1\}$ (boolean logic) is analyzed and a functional completeness result is given for polynotopes. Based on operator overloading, the generic implementation of an original Polynotopic Kalman Filter (PKF) extending Zonotopic Kalman Filters (ZKF) to hybrid nonlinear systems is obtained. Several discrete, continuous and hybrid numerical examples illustrate the main theoretical results.

Organization. After extending the notion of inclusion function classically used in interval arithmetic to general sets in section 2, the motivation and the construction/-composition of polynotope objects is treated in section 3. Uniquely identified typed symbols are also introduced in this section. A possible implementation of the polynotope objects is then gradually introduced in section 4 by first starting from symbolic/mixed zonotopes and mixed encoding of intervals, and then extending sparse polynomial zonotopes with specific features making it possible to handle mixed sets in a unified way. In section 5, modeling tools for nonlinear hybrid systems are given with emphasis placed on a compositional approach relying on basic logic gates and basic nonlinear continuous and switching functions. Inclusion methods are also given. Then, a Polynotopic Kalman Fiter (PKF) extending ZKF to hybrid nonlinear systems is developed in section 6. Through basic operators/functions overloading, its implementation can benefit from the proposed dependency preserving compositional inclusion methods. The links between PKF, ZKF and the basic stochastic Kalman Filter KF [19] are made explicit. In section 7, numerical examples including comparisons illustrate the effectiveness of the proposed scheme, before concluding remarks in section 8.

Table 1
Examples of possible domains $\mathcal{D}x$ and $\mathcal{D}X$ for a variable x

Example	$\mathcal{D}x$	$\mathcal{D}X$
Continuous	$\bar{\mathbb{R}}^n$	$\mathbb{I}\bar{\mathbb{R}}^n$
Discrete	$\{0,1\}$	$\{\{0\},\{1\},\{0,1\}\}$
Mixed	$\{0,1\}\cup[2,4]$	$\{\{1\},\mathbb{I}[2,3],\{0\}\cup[3,4],\mathcal{D}x\}$

2 Inclusion function: beyond intervals

To begin with, a definition of sets from functions (imset) and a definition of inclusion functions are given and discussed in a classical (non-symbolic) framework. Given a variable x, let $\mathcal{D}x$ denote a set (or domain) of possible values for x, that is: $x \in \mathcal{D}x$. Also, let $\mathcal{D}X = {}^{\subset}\mathcal{D}x$ denote a set of subsets of $\mathcal{D}x$, including $\mathcal{D}x$ itself, that is: $(\forall X \in \mathcal{D}X,\, X \subset \mathcal{D}x) \wedge (\mathcal{D}x \in \mathcal{D}X)$. Since the value of any $X \in \mathcal{D}X$ is a set, X is said *set-valued*. Notice that $x \in \mathcal{D}x$ is not necessarily set-valued. For instance, the Table 1 reports some continuous, discrete and mixed examples, where $\bar{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$, and $\bar{\mathbb{I}}X$ (resp. $\mathbb{I}X$) refers to the set (resp. collection) of real intervals $[a,b]$ included in some set X. Thus, $\bar{\mathbb{I}}X = \{\mathbb{I}X\}$. The continuous example with $\mathcal{D}X = \mathbb{I}\bar{\mathbb{R}}^n$ is classically used to define inclusion functions in the particular case of interval arithmetic with possibly unbounded [1] intervals.

Definition 1 (imset) *Given a function $f : \mathcal{D}x \to \mathcal{D}y$, $x \mapsto y = f(x)$, and a set $X \subset \mathcal{D}x$, the imset of X by f is:*
$$f(X) = \{f(x)\,|\, x \in X\}.$$

Definition 2 (Inclusion function) *The function $g : \mathcal{D}X \to \mathcal{D}Y$, $X \mapsto Y = g(X)$ is an inclusion function for $f : \mathcal{D}x \to \mathcal{D}y$, $x \mapsto y = f(x)$, if $\mathcal{D}x \in \mathcal{D}X$ and:*
$$\forall X \in \mathcal{D}X,\; f(X) \subset g(X),$$
where $f(X)$ is the imset of $X \subset \mathcal{D}x$ by f, and $g(X)$ is the image of $X \in \mathcal{D}X$ by g.

Corollary 3 *g must not be monotone wrt inclusion to be an inclusion function. Even without this requirement, it can be inferred that:*
$$\forall X \in \mathcal{D}X,\; f(X) \subset g(\mathcal{D}x).$$

Proof. Firstly, $imset_f : \mathcal{D}X \to \mathcal{D}Y$, $X \mapsto Y = f(X)$ is monotone wrt inclusion: $X_1 \subset X_2 \Rightarrow imset_f(X_1) \subset imset_f(X_2)$. Moreover, from the definition 2, $\forall X \in \mathcal{D}X$, $X \subset \mathcal{D}x$, and the monotony of $imset_f$ wrt inclusion gives: $\forall X \in \mathcal{D}X$, $f(X) \subset f(\mathcal{D}x)$. Also, since $\mathcal{D}x \in \mathcal{D}X$, $f(\mathcal{D}x) \subset g(\mathcal{D}x)$. Thus, $\forall X \in \mathcal{D}X$, $f(X) \subset g(\mathcal{D}x)$, without requiring the additional statement that g must be monotone wrt inclusion to be an inclusion function.

Remark 4 *In general, $2^{\mathcal{D}x} \not\subset \mathcal{D}X$. This may lead to some conservatism when using an inclusion function defined over $\mathcal{D}X$ to enclose the image of any arbitrary subset $X \in 2^{\mathcal{D}x}$ of $\mathcal{D}x$. This kind of wrapping effect is hardly avoidable in practice since, in most cases, not all subsets of $\mathcal{D}x$ can be exactly represented in machine, especially when dealing with continuous and/or hybrid domains.*

Remark 5 *Syntax and semantics are not distinguished in this section 2 where the same notation x may refer to the (name of a) variable or a taken value.*

A basic illustration of interval arithmetic's dependency problem is first given and analyzed. Let x denote any real such that $x \in [-1,+1] \subset \mathbb{R}$. Let $f : \bar{\mathbb{R}} \to \bar{\mathbb{R}}$, $x \mapsto x - x$. *i)* Since $\forall x$, $x - x = 0$, f is the null function and the imset $f([-1,+1])$ of the interval $[-1,+1]$ by f is obviously the singleton $\{0\}$. *ii)* Let g be the natural interval extension of f i.e. $g : \mathbb{I}\bar{\mathbb{R}} \to \mathbb{I}\bar{\mathbb{R}}, x \mapsto x - x$, where the minus operator is "overloaded" to deal with interval operands. g is an inclusion function for f. Thus: $f([-1,+1]) \subset g([-1,+1]) = [-1,+1] - [-1,+1] = [-2,+2]$. Conclusion: Though inclusion is preserved, $[-2,+2]$ is a poor outer approximation of $f([-1,+1]) = \{0\}$. Whereas simply overloading basic interval operators would be the dream of a programmer wishing to implement uncertainty propagation computations, the natural interval extension is often subject to a significant conservatism. This is mainly due to the loss of dependencies between multiple occurrences of variables in the expressions to be evaluated (e.g. notice that x occurs twice in f and g). Indeed, important symbolic links are lost when substituting some interval value (semantics) for a given variable name/symbol (syntax) occurring in expressions/formula used to define mathematical functions and/or imsets as in definition 1. Moreover, iterative evaluations, e.g. to compute reachable sets of dynamical systems, often emphasize the drawbacks of a natural interval extension. Then, should we definitely abandon the dream of encoding guaranteed (inclusion preserving), accurate and fast uncertainty propagation algorithms simply by overloading the operators used to define and program explicit mathematical functions? Beyond the natural interval arithmetic extension, how to tackle the dependency problem while dealing with possibly mixed (hybrid: continuous and discrete), non-convex and non-connected sets in a unified way?

3 Polynotope objects: Why and how?

3.1 Functional sets

Starting from intervals to represent bounding sets over continuous domains, explicit descriptions of more general sets is a possible direction to struggle against the dependency problem. Though general polytopes may look attractive to represent/approximate a large class of convex sets, the enumeration of their vertices and/or faces often remains poorly scalable. Paving and extensive use of contractions and bissections (branching) is another

[1] Note here that $\mathbb{R}^n =]-\infty,+\infty[^n \in \mathcal{D}X$, $\mathcal{D}x = \bar{\mathbb{R}}^n = [-\infty,+\infty]^n \in \mathcal{D}X$, and $\mathcal{D}X = \mathbb{I}\bar{\mathbb{R}}^n$ is a strict subset of the power set $2^{\mathcal{D}x}$ of $\mathcal{D}x = \bar{\mathbb{R}}^n$ (e.g. a sphere is not an interval).

possible direction. However, bissections often restrict the scalability of accurate set approximations as the space dimension increases. Implicit rather than explicit set descriptions have also received a significant attention along the years e.g. ellipsoids described through a symmetric and positive definite (spd) matrix or as the imset of a unit hypersphere by a linear function, zonotopes (resp. polynomial zonotopes) most often viewed/defined as the imset of a unit hypercube $[-1,+1]^p \subset \mathbb{R}^p$ by an affine (resp. polynomial) function $f : \mathbb{R}^p \to \mathbb{R}^n$. For instance, the zonotope $\langle c, R\rangle = \{c + Rs,\ s \in [-1,+1]^p\} \subset \mathbb{R}^n$ is nothing else but the imset of $[-1,+1]^p$ by the function $f_{c,R} : s \mapsto c + Rs$. Though the possible shapes of zonotopes in $\mathbb{R}^n$ with $n > 2$ are significantly more versatile than that of basic nD intervals, so usually providing very useful degrees of freedom to struggle against the wrapping effect, it remains worth noticing the consequences of the intrinsic symbolic (syntactic) *difference* between the two expressions $f_{c,R}(s_a)$ and $f_{c,R}(s_b)$ where, e.g., $s_a \in [-1,+1]^p$ and $s_b \in [-1,+1]^p$. First, the possible values (semantics) taken by both expressions belong to the *same* zonotope $\langle c, R\rangle$. Indeed, the set-based wrap/enclosure $[f_{c,R}(s_a)]$ of $f_{c,R}(s_a)$ is $[f_{c,R}(s_a)] = f_{c,R}([s_a]) = f_{c,R}([-1,+1]^p) = \langle c, R\rangle$ and idem with s_b since $[s_b] = [s_a] = [-1,+1]^p$ (and so, even if $s_a \neq s_b$). In this context, what about the composition of set-based enclosures?

$[f_{c,R}(s_a) + f_{0,-R}(s_a)]\ \ldots$
$a_1)\quad \ldots \subset [c + Rs_a - Rs_a] \subset \{c\} = \langle c, 0\rangle$
$a_2)\quad \ldots \subset [f_{c,R}(s_a)] + [f_{0,-R}(s_a)] \subset \langle c, 2R\rangle$

$[f_{c,R}(s_b) + f_{0,-R}(s_a)]\ \ldots$
$b_1)\quad \ldots \subset [c+Rs_b-Rs_a] \subset \{c\}+R[s_b]-R[s_a] \subset \langle c, 2R\rangle$
$b_2)\quad \ldots \subset [f_{c,R}(s_b)] + [f_{0,-R}(s_a)] \subset \langle c, 2R\rangle$

In the cases a_1 and b_1, a transformation/simplification of the symbolic expressions is first conducted (possibly based on an eager evaluation of terms interpreted as symbols rather than values) and the evaluation of set-based enclosures is delayed as much as possible (lazy evaluation). In the cases a_2 and b_2, an eager evaluation using set-based enclosures as possible values is performed without any prior symbolic transformation. In the case of a_2 and compared to a_1, this second approach yields a significant conservatism originating from the loss of dependencies between the two (thus, multiple) occurrences of s_a which distinguish a_1, a_2 from b_1, b_2.

At this step, several directions orienting the following of this work can be drawn:
i) Functions can be used to implicitly define sets as the image of their definition domain (e.g. unit hypercubes with affine functions $f_{c,R}$ for classical zonotopes).
ii) Working out a traceability of variable multi-occurrences within the (possibly incrementally built) symbolic expressions used to define such functions/sets can help to struggle against the dependency problem.
iii) While elementary operators overloading is preferred to encapsulate the code required to easily compose complex functions/sets from simple ones, a global[2] scope should be maintained for symbolic variables to prevent from losing global dependencies.
iv) A trade-off between the efficiency of symbolic operations, their memory footprint, and the ability to quickly compute guaranteed and accurate set-based enclosures has to be found, possibly involving reduction schemes.
v) By default, binding/linking/sharing is preferred to systematic bissection/branching in this work.
vi) Set-based enclosures should be computed only when needed (call by need), so leading to a lazy/delayed evaluation of wrapping sets and/or bounds.
vii) The tight interactions between symbolic expressions and mathematical definitions as well as numerical computations reveals the need for distinguishing between syntax (symbolic level) and semantics (mathematical interpretations of symbolic expressions possibly resulting from numerical computations).

3.2 Syntax and semantics

With the ultimate goal of better managing dependencies that constitute a key for an accurate propagation of uncertainties, an explicit distinction between syntax and semantics is considered. Whereas syntax refers to rules defining symbol combinations that are correct in some language, semantics refers to the interpretation or meaning of related sentences. In other words, syntax refers to how writing correct statements, semantics indicates what they mean.

Let z denote a (symbolic) name that should be understood/interpreted as an object ιz. The interpretation operator ι assigns a meaning (e.g. mathematical or computational object, set, numerical value, etc) to the symbolic name z (and possibly to other symbols/names). In this paper, ι is a generic notation used to disambiguate symbolic names (syntax) from their meaning and/or taken values (semantics), whenever necessary. ιz can be viewed as a shortcut for $\iota(z)$ or $[\![z]\!]_\iota$ which is a usual notation for valuation (semantics) in logic and computer science. However, systematic formal definitions of interpretation/valuation operators are out of scope in this paper: ι simply reads as "interpretation of" for the sole purpose of drawing an explicit separation between operations at syntactic and semantic levels.

For example, $\mathcal{D}x$ denoting a set containing the possible values (definition domain) of a variable named/symbolized by x. Then, $\iota x \in \mathcal{D}x$ reads as "the interpretation/-value assigned to the variable named x belongs to the definition domain of x". Notice that usual mathematical notations do not distinguish between variable names and values (e.g. when writing $x \in \mathcal{D}x$ for the variable x, as in section 2). Similarly, f may denote the name/symbol

[2] not only at a host level but also at a wider scale to also cover the network nodes of distributed systems like CPS.

referring to a function explicitly denoted as ιf. Then, ιf stands for an interpretation of f (e.g. as a mathematical function, as a subroutine/algorithm, etc). Moreover, an evaluation $\iota f(\iota x)$ of the function named f stands for the image/result obtained by applying ιf to the value ιx assigned to the input variable named x.

3.3 Typed symbols and unique identifiers

The so-called polynotope objects are interpreted (semantics) as image sets of vector polynomial functions defined from symbolic expressions (syntax) on domains related to different types of symbolic variables. For instance, continuous (unit interval) and/or discrete (signed, boolean) symbols can be combined to define and operate possibly mixed/hybrid sets. Following [9], each of these symbols are uniquely identified in order to preserve dependencies while simplifying otherwise possible name binding issues. Considering other basic symbol types than interval (`i`), signed (`s`) and boolean (`b`) is among possible extensions that are left to future works:

Assumption 6 (Basic types) *Let $\mathbb{S}$ be a finite set of typed symbols. Each typed symbol $s_i \in \mathbb{S}$ is uniquely identified by an integer $i \in \mathbb{N}$. In this paper, the type τs_i of any s_i belongs to the set $\mathbb{T} = \{\mathtt{i}, \mathtt{s}, \mathtt{b}\}$ of three basic symbol types respectively referring to interval (`i`), signed (`s`) and boolean (`b`). As shown in table 2, a definition domain is related to each of these basic types as $\square = [-1, +1]$, $|^{+}_{-}| = \{-1, +1\}$, $|^{1}_{0}| = \{0, 1\}$, respectively. By default, basic scalar values are assumed to be interpreted in the real field $\mathbb{R}$ equipped with the usual sum and product operators. A partition of $\mathbb{T}$ into continuous and discrete types is given by $\mathbb{T} = \mathbb{T}_c \cup \mathbb{T}_d$ with $\mathbb{T}_c = \{\mathtt{i}\}$ and $\mathbb{T}_d = \{\mathtt{s}, \mathtt{b}\}$.*

Definition 7 (Mixed, continuous, discrete) *Let $I \subset \mathbb{N}$ and $T_I = \cup_{i \in I}\{\tau s_i\}$. The symbolic vector s_I is mixed (resp. continuous, discrete) if $(T_I \cap \mathbb{T}_c \neq \emptyset) \wedge (T_I \cap \mathbb{T}_d \neq \emptyset)$ (resp. $T_I \subset \mathbb{T}_c$, $T_I \subset \mathbb{T}_d$). By extension, any formula $F(s_I)$ and/or related interpretation as (s-)function, (s-)zonotope, (s-)polynotope, etc can be qualified as mixed, continuous[3] or discrete accordingly.*

Corollary 8 *In an entirely continuous case, following the assumption 6, all the symbols in s_I are of type (unit) interval: $\forall i \in I$, $\tau s_i = \mathtt{i}$, that is, $\forall i \in I$, $\iota\tau s_i = \square = [-1; +1]$. As a result, for any vector I of n unique identifiers only referring to continuous symbols, any single-valued interpretation[4] ιs_I of the symbolic vector s_I belongs to the unit interval $\square^n$, that is, $\iota s_I \in [-1; +1]^n$.*

Assumption 9 (USP) *A Unique Symbol Provider (USP) is assumed to be implemented as a global function (or service in a distributed context) called as $!(n, t)$*

Table 2
Notations for three basic symbol types in $\mathbb{T} = \{\mathtt{i}, \mathtt{s}, \mathtt{b}\}$ respectively referring to "interval", "signed", "boolean".

	syntax	semantics	
typed symbol	$s_i = \mathtt{x{:}i}$	$\iota s_i \in \mathcal{D}s_i$	*interval:*
symbol type	$\tau s_i = \mathtt{i}$	$\iota\tau s_i = \mathcal{D}s_i = \square = [-1, +1]$	
typed symbol	$s_j = \mathtt{z{:}s}$	$\iota s_j \in \mathcal{D}s_j$	*signed:*
symbol type	$\tau s_j = \mathtt{s}$	$\iota\tau s_j = \mathcal{D}s_j = |^{+}_{-}| = \{-1, +1\}$	
typed symbol	$s_k = \mathtt{y{:}b}$	$\iota s_k \in \mathcal{D}s_k$	*boolean:*
symbol type	$\tau s_k = \mathtt{b}$	$\iota\tau s_k = \mathcal{D}s_k = |^{1}_{0}| = \{0, 1\}$	

`x`, `y`, `z` : particular examples of (untyped) symbol names.
The unique identifier of any typed symbol $s_i \in \mathbb{S}$ is $i \in \mathbb{N}$.

with $(n, t) \in \mathbb{N} \times \mathbb{T}$ and then returning an n-dimensional vector $I \in \mathbb{N}^n$ of n new unique identifiers referring to n new typed symbols of type t i.e. $\forall i \in I$, $\tau s_i = t$.

Remark 10 *A simple implementation of $!(n, t)$ is "$l = l + n$, return$(h(t)\mathbf{1}_n + 2^{n_h}[l - n + 1, ..., l])$" where $\mathbf{1}_n$ denotes a vector of n ones, l is a persistent counter initialized to 0 at startup, and $h : \mathbb{T} \to \mathbb{N}$ assigns to each type $t \in \mathbb{T}$ a unique integer tag encoded with at most n_h bits. Under the assumption 6, reserving $n_h = 2$ bits and taking $h(\{\mathtt{i}, \mathtt{s}, \mathtt{b}\}) = \{1, 2, 3\}$ is a possible choice to efficiently manage type tags within the unique integer identifiers of typed symbols. Extensions include an overflow checking or an implementation (possibly distributed) as a service in a CPS (Cyber-Physical System): see [9] for details.*

3.4 Construction/composition of polynotope objects

Based on uniquely identified typed symbols as in 3.3, the remainder of this section 3 aims at describing how the directions listed at the end of 3.1 can be achieved through polynotope objects, while introducing useful definitions to build well-grounded (e.g. inclusion preserving) mathematical interpretations. Consistently with the encapsulation principle [32], internal data structures are left open in this section whereas some choice for them will be made explicit in section 4.
Polynotopes interprets as functional sets based on vector polynomial functions with typed symbols i.e. any polynotopic set can be viewed as the imset of the definition domain of a vector polynomial function depending on variables/symbols of different types such as, e.g., the basic types in assumption 6. Meanwhile, a mechanism compatible with operator overloading is needed to achieve the direction *iii*) in 3.1, that is, providing a simple interface for the user to naturally encode non trivial compositions of elementary polynotopes, while maintaining a) a global scope for typed symbols to struggle against the so-called dependency problem and b) the rigor of inclusion preserving set-based interpretations. For this purpose, a formal grammar describing the syntax of a prototype language is given in Table 3. It supports a basic polynotope composition scheme (mainly

[3] Regarding functions, continuous refers here to a property of the input domain and not to continuity as in analysis.
[4] as long as it is consistent with the type (unit) interval.

Table 3
Context-free grammar in Backus-Naur Form (BNF) describing the syntax of a prototype language supporting the basic construction (through typed symbols) and composition of Polynotope objects.

```
BNF
1   <letter>      ::= "A".."Z" | "a".."z"
2   <digit>       ::= "0".."9"
3   <namechar>    ::= <letter> | <digit> | "_"
4   <name>        ::= <letter> | <name> <namechar>
5   <integer>     ::= <digit> | <digit> <integer>
6
7   <symbol>      ::= <name>
8   <type>        ::= "i" | "b" | "s"   % For instance. See (a)
9   <typedsymbol> ::= <symbol> ":" <type>
10  <variable>    ::= <typedsymbol> | <sfunref>
11
12  <exponent>    ::= <integer>
13  <multerm>     ::= <variable> | <variable> "^" <exponent>
14  <monomial>    ::= <multerm> | <multerm> "*" <monomial>
15  <coefficient> ::= <IEEE 754 floating point number>   % For instance. See (b)
16  <sumterm>     ::= <coefficient> | <monomial> | <coefficient> "*" <monomial>
17  <polynomial>  ::= <sumterm> | <sumterm> "+" <polynomial>
18  <polynlist>   ::= <polynomial> | <polynomial> ";" <polynlist>
19  <polynvector> ::= "[" <polynlist> "]"
20  <polynotope>  ::= <polynomial> | <polynvector>
21
22  <sfunction>   ::= <polynotope>   % For instance. See (c)
23
24  <sfunname>    ::= <name>
25  <sfunindex>   ::= <integer>
26  <sfunref>     ::= <sfunname> | <sfunname> "(" <sfunindex> ")"
27  <sfundecl>    ::= <sfunname> "=" <sfunction>
28  <sfunprog>    ::= <sfundecl> | <sfundecl> "," <sfunprog>
```

(a) Consistently with assumption 6. Polynotopes may support other types.
(b) Efficient numeric computations of constant/center vectors and coefficient/generator matrices of polynotope objects motivate this choice.
(c) At a syntactic level, canonical polynotopes (obtained after resolving <sfunref> references allowing to compose polynotopes) are polynomial vectors with only typed symbols as variables. Notice that vector linear functions (obtained from <monomial> ::= <variable> while removing <exponent>, <multerm>) give symbolic zonotopes and that nothing a priori hinders the definition of other <sfunction> as functions of typed symbols.

Table 4
Left: Sample code accepted by the grammar in Table 3. Right: Evaluation of bounds resulting from a natural interval extension (int.) or some[†] polynotope computations (pol.).
[†] here, e.g., interval hull of an outer zonotope.

	sample code	int.	pol.
1	u=symb:i,	[-1,+1]	[-1,+1]
2	x=0.5+0.5*u,	[0,1]	[0,1]
3	f1=[x*x; x],	[0,1]^2	(*)
4	f2=f1(2)-f1(1),	[-1,1]	[0,1/4]
5	f3=[x^2; x],	[0,1]^2	(*)
6	f4=f1-f3,	[-1,+1]^2	{0}^2
7	r=remainder:i,	[-1,+1]	[-1,+1]
8	f5=[x+-0.125+0.125*r; x],	(*)	(*)
9	f6=f5(2)-f5(1)	[-1,5/4]	[0,1/4]

(*) = [[-1/4,1];[0,1]]

based on the <sfunref> lexical tokens[5]), as exemplified with the sample code in Table 4. Also, polynomials being closed under a finite[6] number of compositions, it describes/accepts a set of well-formed formulas (wff) that can be operated at a symbolic level[7]. Such operations preserve the ability to delay as much as needed set-based evaluations of typed symbols (lazy evaluation), while providing the formal ground for inclusion preserving mathematical interpretations[8]. Moreover, the eager evaluation of an <sfunprog> source code (like the sample in table 4) using polynotope objects allows to transform polynotopic symbolic function (s-function) declarations <sfundecl> (see also note (c) in Table 3) into s-function definitions <sfundef>, that is, a canonical kind of <sfunction> such that all the references <sfunref> to other s-functions have been resolved[9]. Concretely, polynotope objects store (the abstract syntax tree of) vector polynomial functions of typed symbols as sole variables, without any reference <sfunref> to other polynotope objects. <sfunref> references however remain a key enabler to compose non trivial polynotopes (by using intuitive operator overloading) from basic ones, which are typically constructed from <typedsymbol>s as shown at lines 1 and 7 in Table 4. Though an eager evaluation is firstly used to solve <sfunref> tokens *only*, <typedsymbol> tokens are *not* immediately evaluated. This makes it possible to delay their evaluation while keeping trace of *global* dependencies between polynotope objects since each typed symbol is encoded through a unique identifier. By tagging the symbol identifiers with their type, not only the symbols evaluation can be delayed (lazy evaluation) but also adapted to their type (polymorphism). As a result, polynotopes permit some kind of polymorphic delayed evaluation of uncertain symbols/variables. Moreover, a global scope is preserved for the latter, which is the basic key to address the dependency problem arising from a direct use of interval arithmetic. The interplay between mathematical definitions and the semantics attached to the intermediate (polynotope objects) and finally computed values (e.g. zonotopes, intervals) also requires a special attention to ensure inclusion is preserved, not only on continuous domains, but also on discrete and mixed ones, as is made possible with polynotopes.

3.5 s-functions

In order to transform and evaluate expressions based on typed symbols, the notion of (polynotopic) symbolic function or, shortly, s-function, is precised. s-function definitions (sfd), some of their interpretations (sffi) and related evaluations (sffe) are then considered. Whereas sfd refers to syntax, sffi and sffe refer to semantics.

Let $\mathbb{F}$ denote the set of finite length well-formed formulas (wff) corresponding to canonical polynotopic symbolic functions as described by <sfundef> in paragraph 3.4 and, more specifically, in footnote 9. Let $f \in \mathbb{F}$ be such

[5] A (lexical) token is a string (i.e. a sequence of characters) with an assigned and thus identified meaning.

[6] In this paper, a sufficiently large finite context is assumed. Concretely, this can be achieved through inclusion preserving reduction schemes, as in definition 17 for instance.

[7] e.g. by using efficient data structures to encode and manipulate the abstract syntax trees of vector polynomials with typed symbols: see (8) and symbolic polynotopes in 4.3.

[8] e.g. see (9), e-polynotopes and definition 30 in 4.3.

[9] <sfundef> thus stands for an <sfunction> as in Table 3 except that <sfunref> is removed from line/rule 10 which then becomes <variable>::=<typedsymbol>.

a wff. Firstly, f describes a possibly scalar polynomial vector (see lines 20 and 22 in table 3) of finite dimension $n_f \in \mathbb{N}$. Also, f contains a finite number $p_f \in \mathbb{N}$ of typed symbols (`<typedsymbol>` tokens). Let $I = I_f \subset \mathbb{N}$ be the set [10] of their unique identifiers. Then, to emphasize its dependence on the typed symbols in s_I, the wff f can be equivalently [11] denoted as $F(s_I)$ with $f = F(s_I)$.

Definition 11 (s-function: sfd/sffi/sffe) *:*
• *s-function definition (sfd) :* $f = F(s_I) \in \mathbb{F}$. *$F(s_I)$ is a wff involving the (typed) symbolic variables in s_I which become bound in the formula: indeed, $F(s_I)$* depends *on s_I, at least from a syntactical viewpoint.*
• *s-function functional interpretation (sffi) : denoted as $\iota f(.)$ to emphasize its* functional nature, *an sffi of f is an interpretation ιf of f that has the* ability *to define and/or return output values from input values corresponding to an interpretation/valuation of the (typed) symbols in s_I.*
• *s-function functional evaluation (sffe) : Let ιs_I be an interpretation/valuation of the symbolic vector s_I and let $\iota f(.)$ be an sffi of $f = F(s_I)$. Then, the sffe $\iota f(\iota s_I)$ is the* result *obtained by applying ιs_I to the sffi $\iota f(.)$.*

Remark 12 (sffi vs sffe) *An s-function interpretation ιf, as long as it is also an sffi $\iota f(.)$, should be understood as a mathematical function or subroutine or algorithm or any transformation process related to f but* not *as some returned output value obtained from such a function or process. By contrast, an sffe $\iota f(\iota s_I)$ should be understood as some* returned value *obtained from an sffi $\iota f(.)$ fed by some interpretation/valuation ιs_I of the typed symbols s_I referring to its (possibly uncertain) inputs. Though out of scope in this work, note that there is no incompatibility with taking functions as possible values, like in functional paradigms. Moreover, random variables being nothing else but functions from a set of outcomes to a set of possible values, the proposed approach is open to extensions involving stochastic descriptions.*

An s-function definition (sfd) usually results from the eager evaluation of some program containing s-functions declarations like, e.g., `<sfunprog>` in Table 3. For polynotopes, this evaluation is based on a systematic expansion leading to a polynomial vector with variables corresponding to typed symbols only, and no more reference to any other s-functions which are all resolved. In other words, the systematic polynomial expansion gives a fully unfolded abstract syntax tree, up to typed symbols i.e. s-function uncertain inputs. For example, based on the sample code in lines 1-4 of Table 4, the following s-function definitions (sfd) are obtained:

$f_1 =$ `[0.25+0.5*symb:i+0.25*symb:i 2; 0.5+0.5*symb:i]`,
$f_2 =$ `0.25-0.25*symb:i 2`.

[10] or a vector. Here, the ordering of scalar elements is free.

[11] Note that, alternately, $F(s_I)$ (resp. $F(.)$) may refer to the abstract syntax tree resulting from parsing the wff f up to typed symbol leafs included (resp. not included).

Note that the elimination of `0.5*symb:i` in the sfd of f_2 is made possible at the symbolic level. This greatly helps to improve the evaluation of some bounds in Table 4.

Several interpretations of an s-function $f = F(s_I)$ may coexist but a basic one is as a mathematical function like

$$\iota f : \iota\tau s_I \to \mathbb{R}^{n_f},\ \iota s_I \mapsto \iota f(\iota s_I), \tag{1}$$

which is an sffi. It is important to note in (1) that the definition domain $\iota\tau s_I$ of ιf depends on the types τs_I of the typed symbols s_I involved in the wff $F(s_I)$ defining the s-function $f \in \mathbb{F}$. See also Table 2. Continuing the example, `symb:i` being of type `i` (interval), it comes:

$\iota f_1 : [-1,+1] \to \mathbb{R}^2,\ \sigma \mapsto [0.25 + 0.5\sigma + 0.25\sigma^2; 0.5 + 0.5\sigma]$,
$\iota f_2 : [-1,+1] \to \mathbb{R},\ \sigma \mapsto 0.25 - 0.25\sigma^2$.

An hybrid example mixing continuous and discrete types is also given with a s-function $f \in \mathbb{F}$ defined as the wff $F(s_I) =$ `1+x:i+4*y:b*z:s`. Then, $n_f = 1$ (scalar output), and the $p_f = 3$ involved typed symbols $s_{I_1} =$ `x:i`, $s_{I_2} =$ `y:b`, $s_{I_3} =$ `z:s` are respectively of type `i` (interval), `b` (boolean), `s` (signed). In this example, the symbol names in $F(s_I)$ coincide with those in Table 2. Following (1), an sffi of f as a mathematical function is:

$\iota f : \square \times |_0^1| \times |_-^+| \to \mathbb{R},\ [x; y; z] \mapsto 1 + x + 4yz$,

and the imset of $\iota\tau s_I$ by ιf is $[-4,-2] \cup [0,2] \cup [4,6]$ which is a non-convex and non-connected set. This can be generalized with image-sets.

3.6 Image-sets

Set-based rather than functional (sffi) interpretations of s-functions are considered with the image-sets defined in this paragraph. They provide set-based wraps ensuring the consistency between some intended mathematical meaning and the actually computed sets through an inclusion preserving approach. This is obtained by first extending the *imsets* and *inclusion functions* in section 2 to a symbolic context with typed symbols through *image-sets* and *inclusion s-functions*, respectively. Then, a notion of inclusion preserving symbolic reduction operator is introduced. This gives control to maintain finite representations of prescribed complexity for the underlying approximate polynomial expansions while preserving a set inclusion property ensuring consistency of the computed polynotopic image-sets.

Definition 13 (Image-set) *The image-set $\langle f \rangle_\iota$ of the s-function $f = F(s_I) \in \mathbb{F}$ is the imset of the domain $\iota\tau s_I$ by a functional interpretation ιf of f:*

$$\langle f \rangle_\iota = \{\iota f(\sigma) \mid \sigma \in \iota\tau s_I\} = \iota f(\iota\tau s_I).$$

Remark 14 *The domain $\iota\tau s_I$ is a set related to the types of the symbols in s_I (see assumption 6 and table 2) and $\iota f(\sigma)$ stands for an sffe as in definition 11.*

Definition 15 (Inclusion s-function) *The s-function $g = G(s_J)$ is an inclusion s-function for $f = F(s_I)$ under functional interpretations ιf of f and ιg of g if ιg is an inclusion function for ιf defined on the domain $\iota\tau s_I$.*

Corollary 16 *From the definition 2 and its corollary 3, ${}^{\subset}\iota\tau s_I$ being a set of subsets of $\iota\tau s_I$ including $\iota\tau s_I$ itself, $\iota f(\Sigma)$ being the imset of Σ by ιf, and $\iota g(\Sigma)$ being the image of Σ by ιg, it comes:*

$$\forall\Sigma \in {}^{\subset}\iota\tau s_I,\ \{\iota f(\sigma) \mid \sigma \in \Sigma\} = \iota f(\Sigma) \subset \iota g(\Sigma), \quad (2)$$
$$\forall\Sigma \in {}^{\subset}\iota\tau s_I,\ \iota f(\Sigma) \subset \iota g(\iota\tau s_I). \quad (3)$$

Definition 17 (Reduction) *A reduction is an operator $\downarrow_q$ transforming an s-function $f = F(s_I)$ into an s-function $\bar{f} = \downarrow_q f = \bar{F}(s_{\bar{I}})$ such that $\bar{f}$ is an inclusion s-function for f depending on at most q generator symbols: $card(\bar{I}) \leq q \in \mathbb{N}$ and $card(.)$ gives the cardinal.*

Remark 18 (Reduction) *$\bar{I} \cap I \neq \emptyset$ is not mandatory but often useful to limit the inclusion conservatism while controlling the complexity of $\bar{f}$ through its input dimension. In other words, a reduction operator should preserve the more important symbols/dependencies i.e. the ones which significantly contribute to shaping the graph of the mathematical function symbolized by the s-function f.*

Example 19 (Inclusion s-function) *In the sample code of Table 4, the step 8 declares the s-function f_5 which is an inclusion s-function for the s-function f_1 (resp. f_3) declared at step 3 (resp. step 5). Subsequently, f_6 (step 9) is also an inclusion s-function for f_2 (step 4).*

The example 19 illustrates how (vector) polynomial s-functions like f_1 or f_3 can be rewritten as (vector) linear ones like f_5 while preserving the inclusion property of related set-based interpretations as image-sets. This can even be done automatically by following a general scheme similar to automatic differentiation which makes systematic use of basic operator overloading. Pushing further this idea, the graph of non-polynomial functions, possibly including switching ones, can be enclosed within polynotopes, as it will be further addressed in section 5. This approach contributes to significantly enlarge the class of dynamical systems for which reachability and filtering algorithms (like PKF in section 6) can be readily implemented right from the model equations, just by composing polynotope objects *i*) through intuitive operator/function overloading and *ii*) while preserving the efficiency of uncertainty propagation computations possibly involving continuous, discrete or mixed image-sets.

4 From symbolic zonotopes and mixed encoding to polynotopes

The description of specific data structures and algorithms actually implementing methods related to polynotope objects was left opened to a large extent in section 3, consistently with the encapsulation principle. Such descriptions are the main subject of this section 4. They are gradually introduced by first dealing with symbolic zonotopes, then mixed encoding, before addressing the more general case of polynotopes.

4.1 Affine s-functions and zonotopes

Definition 20 (Affine/linear wff) *The wff $F(s_I)$ is affine in s_I if it can be written as $c + Rs_I$ where the vector c and the matrix R do not depend on the symbolic variables in s_I. In particular, it is linear when c is null or can be omitted i.e. $F(s_I) = Rs_I$. Shortly,*
Affine wff: $F(s_I) = c + Rs_I$.

Definition 21 (s-zonotope) *A symbolic zonotope (s-zonotope) $\langle f\rangle_{s,\tau}$ is an s-function $f = F(s_I)$ such that the wff $F(s_I)$ is affine in the symbolic variables in s_I.*

Definition 22 (e-zonotope) *The e-zonotope related to the s-zonotope $\langle f\rangle_{s,\tau}$ is the image-set $\langle f\rangle_{s,\tau,\iota}$ of $f = \langle f\rangle_{s,\tau}$ under an affine interpretation ιf of f. An e-zonotope is thus a set-valued evaluation (semantics) related to a given s-zonotope (syntax).*

One possible data structure to store a symbolic zonotope is (c, R, I). The related s-function defined by a wff denoted $\langle c, R, I\rangle_{s,\tau}$ is $f = c + Rs_I$, and the related e-zonotope is in (5):

$$\langle c, R, I\rangle_{s,\tau} = c + Rs_I \quad (syntax) \quad (4)$$
$$\langle c, R, I\rangle_{s,\tau,\iota} = \{c + R\sigma \mid \sigma \in \iota\tau s_I\} \quad (semantics) \quad (5)$$

The main differences with classical zonotopes defined as $\langle c, R\rangle = \{c + Rs \mid s \in [-1, +1]^n\}$ are twofold:

Firstly, the interplay between syntax and semantics is not caught by the classical definition, whereas it plays a key role in the management of the so-called dependency problem. For example, assuming an entirely continuous case i.e. all the symbols s_i are of type (unit) interval, let us consider the sum S (resp. Minkowski sum S_ι) of the s-zonotopes $\langle 0, 1, 1\rangle_{s,\tau}$ and $\langle 0, -1, 1\rangle_{s,\tau}$ (resp. $\langle 0, 1, 1\rangle_{s,\tau,\iota}$ and $\langle 0, -1, 1\rangle_{s,\tau,\iota}$). Then, $S = 0 + s_1 - s_1 = 0$ (resp. $S_\iota = \square + \square = 2\square = [-2, +2]$), and the set-valued interpretation $\iota S = \{0\}$ of S is much less conservative than $S_\iota = [-2, +2]$ while preserving the inclusion property under the considered semantics. Indeed, some uncertainty cancellation has been made possible by formal/symbolic transformations respecting operators syntactic rules, whereas this is no more possible through the Minkowski sum following a set-valued evaluation. One strategy to improve accuracy thus consists in delaying such set-valued evaluations. Moreover, whereas $\forall i \in \mathbb{N}$, $\langle 0, 1, i\rangle_{s,\tau,\iota} = \langle 0, 1\rangle = [-1; +1]$, $\langle 0, 1, i\rangle_{s,\tau} = s_i$ is distinct from $\langle 0, 1, j\rangle_{s,\tau} = s_j$ as long as $i \neq j$.

From a computational perspective, Matrices with Labelled Columns (MLC) featuring a column-wise sparsity as first introduced in [9] lead to efficient implementations of s-zonotope operators such as sum, linear image, interval/box hull, etc. The reader is referred to [9] for a detailed description, especially in the sections 4 "Matrices with Labeled Columns (MLC)" and 6 "Symbolic zonotopes". Notice that the definition of symbolic zonotopes in [9] only considers one type of symbols interpreted as random variables with support in the unit interval $[-1,+1]$. This outlines how the approach described in section 3 can also be used in a stochastic paradigm: Indeed, it suffices to consider other types of symbols interpreted as random variables (which are themselves functions, so emphasizing the relevance of cross-connections with functional paradigms). In order to give a flavor about MLC, an informal definition and a sum example are provided. An MLC $M^{|I}$ is a pair (M, I^T) where M is an $n \times p$ matrix and $I \in \mathbb{N}^p$ is a vector such that each scalar I_j for $j = 1, \dots, p$, uniquely identifies the jth column $M_{:,j}$ of M. Shortly, the column of $M^{|I}$ labeled as I_j refers to $M_{:,j}$. An example illustrating the sum of two MLC, $M^{|I} + N^{|J} = P^{|K}$ is:

$$\left[\begin{array}{ccc} 2 & 1 & 5 \\ \hline 1 & 0 & 1 \\ 0 & 1 & 1 \end{array}\right] + \left[\begin{array}{cccc} 3 & 5 & 2 & 8 \\ \hline 2 & 0 & 4 & 6 \\ 0 & 3 & 5 & 7 \end{array}\right] = \left[\begin{array}{ccccc} 1 & 2 & 5 & 3 & 8 \\ \hline 0 & 5 & 1 & 2 & 6 \\ 1 & 5 & 4 & 0 & 7 \end{array}\right]. \tag{6}$$

$$\begin{bmatrix} 0 & 5 & 1 & 2 & 6 \\ 1 & 5 & 4 & 0 & 7 \end{bmatrix} \begin{bmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \\ \mathbf{s}_5 \\ \mathbf{s}_3 \\ \mathbf{s}_8 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{s}_2 \\ \mathbf{s}_1 \\ \mathbf{s}_5 \end{bmatrix} + \begin{bmatrix} 2 & 0 & 4 & 6 \\ 0 & 3 & 5 & 7 \end{bmatrix} \begin{bmatrix} \mathbf{s}_3 \\ \mathbf{s}_5 \\ \mathbf{s}_2 \\ \mathbf{s}_8 \end{bmatrix} \tag{7}$$

As shown in (6), an equal number of columns/generators of the operands is not mandatory, and the labels are reported on the first lines (e.g. $I^T = [2, 1, 5]$). (7) illustrates the column-wise (not row-wise) sparsity granted by MLC operators, and that the sum of two MLC gives the exact sum of two centered s-zonotopes since $\langle 0, P, K \rangle_{s,\tau} = \langle 0, M, I \rangle_{s,\tau} + \langle 0, N, J \rangle_{s,\tau}$. Indeed, $M^{|I} + N^{|J} = P^{|K} \Rightarrow Ms_I + Ns_J = Ps_K$. $K \subset I \cup J$ results from merging the unique identifiers in I and J while removing those possibly related to null generators. The vertical concatenation $[M^{|I}; N^{|J}] = [M^{|I}; 0] + [0; N^{|J}]$ also illustrates close links between sum and concatenation of MLC.

Secondly, another main difference with classical zonotopes is the introduction of symbol typing. This makes it possible to combine several kinds of interpretations. Following the assumption 6, this paper mainly focuses on mixed, continuous and discrete values in a set-membership paradigm, though the approach described in section 3 is more general. For example, let us consider five symbols s_i, $i = 1, \dots, 5$, such that s_1, s_2, s_3 are of type `s` (signed) i.e. s_1, s_2, s_3 take their values

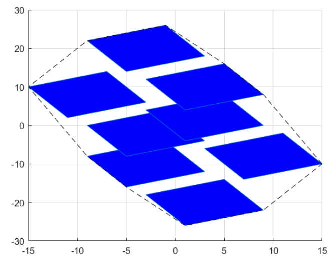

Fig. 1. Example of mixed e-zonotope $\langle 0, R, I \rangle_{s,\tau,\iota}$.

in the discrete set $\{-1,+1\}$, and s_4, s_5 are of type `i` (interval) i.e. s_4, s_5 take their values in the unit interval $[-1,+1] \subset \mathbb{R}$. Let $I_1 = [1, 2, 3]$ and $I_2 = [4, 5]$ be vectors of unique identifiers gathering symbols according to their type: Following the definition 7, s_{I_1} is discrete, s_{I_2} is continuous, and s_I with $I = [I_1; I_2]$ is mixed. Let $R_1 = [3, 3, -3; 6, -5, 9]$, $R_2 = [4, 2; 2, -4]$ and $R = [R_1, R_2]$. Then, the (discrete) e-zonotope $\langle 0, R_1, I_1 \rangle_{s,\tau,\iota}$ is a set of eight [12] points which are the linear images by R_1 of the vertices of a 3D unit hypercube since $\iota s_{I_1} \in \iota\tau s_{I_1} = \{-1,+1\}^3$. The (continuous) e-zonotope $\langle 0, R_2, I_2 \rangle_{s,\tau,\iota}$ is the classical zonotope $\langle 0, R_2 \rangle$ i.e. the linear image by R_2 of a 2D unit hypercube since $\iota s_{I_2} \in \iota\tau s_{I_2} = [-1,+1]^2$. More interestingly, the (mixed) e-zonotope $\langle 0, R, I \rangle_{s,\tau,\iota}$ is the linear image by $R = [R_1, R_2]$ of $\iota\tau s_I = [\{-1,+1\}^3; [-1;+1]^2]$. It is also the Minkowski sum of the (discrete) e-zonotope $\langle 0, R_1, I_1 \rangle_{s,\tau,\iota}$ and the (continuous) e-zonotope $\langle 0, R_2, I_2 \rangle_{s,\tau,\iota}$. The resulting set is neither convex nor connected, as shown in Fig. 1, where the dashed line is the border of the classical (continuous) zonotope $\langle 0, R \rangle$. This example illustrates that mixed zonotopes can provide a very compact representation (Fig.1: $R \in \mathbb{R}^{2\times 5}$ and 5 bits encoding the symbol types) for the union of a same [13] (continuous) zonotopic shape centered on each point of a discrete set possibly containing a high number of configurations (Fig. 1: $2^3 = 8 =$ cardinal of $\{-1,+1\}^3$).

4.2 *Mixed encoding*

The notion of *mixed encoding* is introduced in the same spirit as the example illustrated in Fig. 1. It also provides an approach for a hierarchical modeling of dependencies making it possible to tune the granularity level of the description. Notation: Let $\square_i$ (resp. $|^{+}_{-}|_i$, $|^{1}_{0}|_i$) denote the symbol s_i provided it is of type interval (resp. signed, boolean) i.e. $\tau s_i =$ `i` (resp. `s`, `b`). Under the assumption 6, a compact notation for typed symbols is so obtained, each being uniquely identified by i. Also, let $\rho(0) = 1$, $\rho(n+1) = \frac{1}{2}[1, \rho(n)]$ for $n \in \mathbb{N}$. Then, $\forall n$,

[12] It could be less if some image points are identical.

[13] Relaxing this constraint is among the motivations to the polynomial extension in §4.3.

$\rho(n) = [(\frac{1}{2})^1, \ldots, (\frac{1}{2})^n, (\frac{1}{2})^n] \in \mathbb{R}^{1\times(n+1)}$. By induction, the row sum of $\rho(n)$ is 1.

Definition 23 (Mixed encoding of basic intervals) *The s-zonotope $Z^n_s(I)$ is an n-level signed-interval mixed encoding of the unit interval $[-1,+1]$ if*

$$Z^n_s(I) = \langle 0, \rho(n), I\rangle_{s,\tau} = (\textstyle\sum_{j=1}^n (\frac{1}{2})^j |^{+}_{-}|_{I_j}) + (\frac{1}{2})^n \square_{I_{n+1}}.$$

The s-zonotope $Z^n_b(I)$ is an n-level boolean-interval mixed encoding of the interval $[0,1]$ if

$$Z^n_b(I) = \langle 0, \rho(n), I\rangle_{s,\tau} = (\textstyle\sum_{j=1}^n (\frac{1}{2})^j |^{1}_{0}|_{I_j}) + (\frac{1}{2})^n \square_{I_{n+1}}.$$

Corollary 24 (Mixed encoding of intervals) *Let $Z^n_s(I)$ be a mixed encoding of $[-1,+1]$. Then, $c + rZ^n_s(I)$ is a mixed encoding of $c \pm r = [c-r, c+r]$.*
Let $Z^n_b(I)$ be a mixed encoding of $[0,1]$. Then, $a + (b-a)Z^n_b(I)$ is a mixed encoding of $[a,b]$.

Corollary 25 (Related e-zonotopes) *Following the definition 23, the e-zonotope related to the s-zonotope $Z^n_s(I)$ (resp. $Z^n_b(I)$) is $[-1,+1]$ (resp. $[0,1]$). Conversely, there is no unique mixed encoding for a given interval.*
The e-zonotope $(c + rZ^n_s(I))_\iota$ related to the s-zonotope $c + rZ^n_s(I)$ satisfies $(c + rZ^n_s(I))_\iota \subseteq (c \pm r)$ (interval hull). The equality holds in the scalar case or for $r = 0$.

The surjective nature of mixed encoding gives freedom degrees to model dependencies in a hierarchical way. The discrete parts feature close analogies with the usual binary encoding of integers. Moreover, the coverage of continuous domains is achieved through remainder terms. The width of the set-valued interpretation of these terms is related to the granularity of the mixed-encoding. It can be refined or reduced by adapting the level value n. Since affine s-functions and zonotopes essentially provide operators managing affine dependencies only, and since this yields some restrictions on the possible uses of mixed encoding (among others: see, e.g., footnote 13), an extension to polynomial dependencies is considered.

4.3 Polynomial s-functions and polynotopes

Let $(\boxplus, \boxtimes)$ denote a generic matrix product: $M(\boxplus, \boxtimes)N = \boxplus_{j=1}^p (M_{ij} \boxtimes N_{jk})$, where p both refers to the number of columns of M and the number of rows of N. For example, $MN = M(+,.)N$ is the classical matrix product. $+$, $.$, $\hat{}$ respectively denote sum, product, power.

Definition 26 (Monomial matrix notation) *The monomial matrix θ^E is $(\theta^T(.,\hat{})E)^T$, where θ (resp. E) is a so-called variable matrix (resp. exponent matrix) of dimension compatible with the generic matrix product $(.,\hat{})$. The operator ^{T} refers to transposition.*

Examples: Taking $\theta = [s_1; s_2]$ and $E = [1,0,2;0,3,4]$ yields $\theta^E = [s_1; s_2^3; s_1^2 s_2^4]$. $\theta^{\mathcal{I}} = \theta$ with $\mathcal{I}$ = identity.

Definition 27 (Polynomial wff) *The wff $F(s_I)$ is polynomial in s_I if it can be written as $c + Rs_I^E$ where the vector c and the matrices R and E do not depend on the symbolic variables in s_I. Shortly,*
Polynomial wff: $F(s_I) = c + Rs_I^E$.

Then, c, R, s_I, E, s_I^E are respectively the so-called constant vector, coefficient/generator matrix, symbol(ic variable) vector, exponent matrix, monomial vector.

Definition 28 (s-polynotope) *A symbolic polynotope (s-polynotope) $\langle f\rangle_{s,\tau}$ is an s-function $f = F(s_I)$ such that the wff $F(s_I)$ is polynomial in the symbolic variables in s_I.*

Definition 29 (e-polynotope) *The e-polynotope related to the s-polynotope $\langle f\rangle_{s,\tau}$ is the image-set $\langle f\rangle_{s,\tau,\iota}$ of $f = \langle f\rangle_{s,\tau}$ under a polynomial interpretation ιf of f. An e-polynotope is thus a set-valued evaluation (semantics) related to a given s-polynotope (syntax).*

The name polynotope introduced in this work originates from a contraction of polynomial and zonotope. Following [20], it is also willingly close to polytope. So, *polynotope* gathers, at least partially, the Greek roots of polynomial (from *polus*:numerous and *nomos*:division) and polytope (from *polus* and *topos*:location).
Under the assumption 6, polynotopes are to sparse polynomial zonotopes what mixed zonotopes are to zonotopes. They can also be viewed as constrained polynomial zonotopes with polynomial constraints managed through a possibly extensible set of symbol types.

One possible data structure to store a symbolic polynotope is (c, R, I, E). The related s-function defined by a wff denoted $\langle c, R, I, E\rangle_{s,\tau}$ is $f = (I, c + Rs_I^E)$, and the related e-polynotope is in (9):

$$\langle c, R, I, E\rangle_{s,\tau} = c + Rs_I^E \qquad (syntax) \qquad (8)$$

$$\langle c, R, I, E\rangle_{s,\tau,\iota} = \{c + R\sigma^E \mid \sigma \in \iota\tau s_I\} \; (semantics) \qquad (9)$$

Polynotopes as in (8)−(9) generalize zonotopes as in (4)−(5). Indeed, zonotopes are obtained for $E = \mathcal{I}$ (identity matrix) which is highly sparse and can thus be stored very efficiently. Using a sparse E with integer entries in $\mathbb{N}$ leads to a data structure similar to sparse polynomial zonotopes (spz) in [22], where no typing of symbols is considered (continuous case only). The sparsity of E extends the column-wise sparsity of MLC to polynomial (rather than affine) dependencies with a compact description of monomials featuring (almost) no restriction on the highest degree. The example (10)−(11) shows how the s-polynotope related to (11) can be compactly

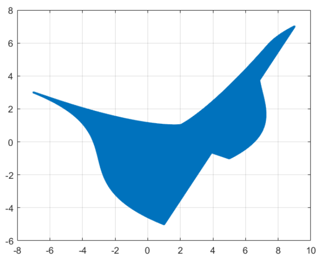

Fig. 2. Example of continuous e-polynotope $\langle c, R, I, E\rangle_{s,\tau,\iota}$.

encoded using (c, R, I, E) as in (10). See also Fig. 2.

$$\left[\begin{array}{c|c} I & E \\ \hline c & R \end{array}\right] = \left[\begin{array}{c|ccc} 1 & 1 & 0 & 1 \\ 8 & 0 & 3 & 1 \\ \hline 2 & 5 & 3 & -1 \\ 1 & 2 & 0 & 4 \end{array}\right], \tag{10}$$

$$\begin{bmatrix} s_1 \\ s_8 \end{bmatrix} \mapsto \begin{bmatrix} 2 + 5s_1 + 3s_8^3 - s_1 s_8 \\ 1 + 2s_1 + 4s_1 s_8 \end{bmatrix}. \tag{11}$$

The implementation of continuous polynotopes operations used in this work is close to the one described in [22] for spz. In particular, each time monomial redundancies might occur, they are removed by summing the related generators: all the columns of E remain distinct. The main differences are:
1) The case of independent generators is not treated separately i.e. all the generators are possibly dependent (provided they share some common symbol),
2) The implementation of a symbolic addition is considered and optimized by taking into account the fact that monomials/generators involving at least one own variable from an operand can be simply copied in the result since no similar monomial exists in the other operand,
3) A vertical concatenation extends the one of MLC,
4) An element-wise product is used as a special case of quadratic map and the reduction extends the one in [9].

Our implementation of polynotopes also supports discrete and mixed operations through symbol typing as described in section 3 and assumption 6. Compared to a strictly continuous case as in [22], the main difference is the introduction of rewriting rules taking the specific nature of signed and boolean symbols into account. Related substitutions ($\rightarrow$) are implemented very efficiently using the (c, R, I, E) attributes with sparse E, e.g.

$$(|^{+}_{-}|_i)^n \rightarrow (|^{+}_{-}|_i)^{mod(n,2)}, \tag{12}$$

$$(|^{1}_{0}|_i)^n \rightarrow (|^{1}_{0}|_i)^{max(n,1)}. \tag{13}$$

Definition 30 (Rewriting rules and inclusion) *A rewriting rule is inclusion preserving if:*

$$(\langle f\rangle_{s,\tau} \rightarrow \langle g\rangle_{s,\tau}) \Rightarrow (\langle f\rangle_{s,\tau,\iota} \subseteq \langle g\rangle_{s,\tau,\iota}).$$

It is inclusion neutral if:

$$(\langle f\rangle_{s,\tau} \rightarrow \langle g\rangle_{s,\tau}) \Rightarrow (\langle f\rangle_{s,\tau,\iota} = \langle g\rangle_{s,\tau,\iota}).$$

Proposition 31 *The rewriting rules in (12) and (13) are inclusion neutral under the assumption 6.*

Notice the syntactical (resp. semantic) nature of the premises (resp. conclusions) of the implications ($\Rightarrow$) in the definition 30. To give insight into the proposition 31, let x be a possible value of any signed symbol: $x \in \{-1, +1\} \subset \mathbb{R}$. Thus, $(x+1)(x-1) = 0$ i.e. $x^2 = 1$ as polynomial constraint. By induction, $x^n = x$ for odd n, $x^n = 1$ for even n, that is $x^n = x^{mod(n,2)}$ which shows the inclusion neutrality of (12). Similarly, let x be a possible value of any boolean symbol: $x \in \{0, 1\} \subset \mathbb{R}$. Thus, $(x-0)(x-1) = 0$ i.e. $x^2 = x$. By induction, $x^n = x$ if $n > 1$, $x^n = 1$ if $n = 0$, that is $x^n = x^{max(n,1)}$ which shows the inclusion neutrality of (13).
The rewriting rules (12)$-$(13) apply for operations modifying the monomial degrees like product; the number of distinct monomials induced by discrete operations is drastically reduced compared to continuous ones, since the exponent in the right term of (12)$-$(13) is either 0 or 1 instead of any $n \in \mathbb{N}$. Thanks to inclusion neutrality, such simplifications of formal expressions induce no conservatism in the related set-valued interpretations. Other rewriting rules are only inclusion preserving:

Proposition 32 *The rewriting rules in (14), (15) and (16) are inclusion preserving under the assumption 6.*

$$(|^{1}_{0}|_i) \rightarrow 1/2 + (|^{+}_{-}|_j)/2, \tag{14}$$

$$(|^{+}_{-}|_i) \rightarrow (\Box_j), \tag{15}$$

$$(\Box_i)^2 \rightarrow 1/2 + (\Box_j)/2, \quad (\Box_i)(\Box_j) \rightarrow (\Box_k). \tag{16}$$

(14)$-$(16) apply before computing the zonotope/interval enclosure of a (possibly mixed) polynotope. Notice that (14) is inclusion neutral if applied globally i.e. without generating new symbol multi-occurrences. (15) is the formal/syntactical counterpart of $\{-1, +1\} \subset [-1, +1]$ which can be viewed as a prototype of the most basic inclusion of two discrete modes/configurations (-1 and $+1$) into a single continuous domain (the unit interval). By using appropriate reductions of formal expressions based on inclusion preserving rewriting rules, mixed polynotopes provide a highly versatile, scalable and computationally efficient approach to combine and enclose possibly non convex and non connected sets under dependency constraints. Hence, they look appealing to deal with verification and synthesis of Cyber-Physical Systems (CPS) whose modeling often relies on mixed/hybrid dynamics. They also exemplify the generality of the approach of image sets with typed symbols described in section 3.

5 Modeling tools for nonlinear hybrid systems

5.1 Discrete: Signed and Boolean logic functions

This paragraph shows how signed (resp. boolean) symbolic variables can be used in a polynomial framework, like the one of polynotopes under the assumption 6, to express any propositional logic formula where symbolic variables are interpreted on a bi-valued real domain: $|^{+}_{-}| = \{-1, +1\} \subset \mathbb{R}$ (resp. $|^{1}_{0}| = \{0, 1\} \subset \mathbb{R}$). This gives a natural interface between continuous variables (defined on a (real) domain with infinite cardinal) and discrete ones (defined on a (real) domain with finite cardinal).

Proposition 33 (Multi-affine decomposition) *Let $f : \mathbb{R}^p \to \mathbb{R}^n$ be any function between finite dimensional real domains. Let $x \in \mathbb{R}$ and $z \in \mathbb{R}^{p-1}$ so that $(x, z) \in \mathbb{R}^p$ refers to any input vector of f where a scalar input x is distinguished from the others. Let introduce four partial functions of f defined as:*

$$
\begin{aligned}
f^A_x(z) &= \tfrac{f(+1,z)+f(-1,z)}{2} && : \textit{Average of } f \textit{ wrt } x,\\
f^H_x(z) &= \tfrac{f(+1,z)-f(-1,z)}{2} && : \textit{Half-gap of } f \textit{ wrt } x,\\
f^G_x(z) &= f(0, z) && : \textit{Ground of } f \textit{ wrt } x,\\
f^U_x(z) &= f(1, z) - f(0, z) && : \textit{Unit-gap of } f \textit{ wrt } x,
\end{aligned}
$$

Then, an affine decomposition of f wrt x under a signed (resp. boolean) x is respectively given by (17) and (18). Moreover, if all the scalar entries of z are signed (resp. boolean), a recursive application of (17) (resp. (18)) results in a (polynomial) multi-affine decomposition of f.

$$x \in \{-1, +1\} \Rightarrow f(x, z) = f^A_x(z) + x f^H_x(z), \qquad (17)$$
$$x \in \{\ \ 0,\ \ 1\} \Rightarrow f(x, z) = f^G_x(z) + x f^U_x(z). \qquad (18)$$

Proof. (17) comes from $f(+1, z) = f^A_x(z) + f^H_x(z)$ and $f(-1, z) = f^A_x(z) - f^H_x(z)$. Similarly, (18) comes from $f(0, z) = f^G_x(z)$ and $f(1, z) = f^G_x(z) + f^U_x(z)$. □

The multi-affine decomposition of basic propositional logic operators is reported in Table 5 both in the signed and boolean cases. At least three noticeable facts emerge from Table 5:

a) The equivalence *eqv* in the signed case features the same multi-affine decomposition as the logical *and* in the boolean case, and both reduce to a simple product.

b) The multi-affine decompositions with signed operands look more "balanced" in terms of involved monomials, compared to the boolean case. This is visible right from the basic affine decompositions in (17) and (18). Indeed, the average of both alternatives (resp. the 0 alternative) serve as reference to express the impact of a switching controlled by x in the signed (resp. boolean) case.

c) Interpreting in $\mathbb{R}$ the polynomial expression of a multi-affine decomposition yields some interpolation between discrete configurations initially expressed in a (bi-valued) propositional logic framework.

Table 5
Signed and Boolean logic functions related to basic operators expressed in the ring of multivariate polynomials $\mathbb{R}[s_I]$ with coefficients in the real field $(\mathbb{R}, +, .)$.

	(a, b)	$\in \{-1, +1\}^2$	$\in \{0, 1\}^2$
Op.	Symb.	Signed	Boolean
not	$\neg$	$-a$	$1 - a$
and	$\wedge$	$\frac{-1+a+b+ab}{2}$	ab
or	$\vee$	$\frac{+1+a+b-ab}{2}$	$a + b - ab$
nand	$\uparrow, \bar{\wedge}$	$\frac{+1-a-b-ab}{2}$	$1 - ab$
nor	$\downarrow, \underline{\vee}$	$\frac{-1-a-b+ab}{2}$	$1 - a - b + ab$
imp	$\Rightarrow, \leq$	$\frac{+1-a+b+ab}{2}$	$1 - a + ab$
eqv	$\Leftrightarrow, =$	ab	$1 - a - b + 2ab$
xor	$\not\Leftrightarrow, \neq$	$-ab$	$a + b - 2ab$
pow	$a^n, n \in \mathbb{N}$	$a^{mod(n,2)}$	$a^{max(n,1)}$
true	$\top$	$+1$	1
false	$\perp$	-1	0

Proposition 34 (Logical ordering) *Let (a, b) be a pair of signed (resp. boolean) symbolic variables. Defining the operator $>$ such that $(a > b) = \neg(a \leq b)$ holds true with operators as in table 5, then $(a < b) = (b > a)$ and $(a \leq b) = ((a < b) \vee (a = b))$ also hold true. More generally, the operators $\leq$ (i.e. implication*[14] *), $\geq, >, <$ follow similar rules as classical order relation operators over reals when signed (resp. boolean) symbols are interpreted with values in $\{-1, +1\} \subset \mathbb{R}$ (resp. $\{0, 1\} \subset \mathbb{R}$).*

Theorem 35 (Functional completeness) *Under the assumption 6, let $I \subset \mathbb{N}$ be a finite set of unique symbol identifiers with at least p elements of type signed (resp. boolean). s-polynotopes based on wff interpreted as multivariate polynomials $\mathbb{R}[s_I]$ with coefficients in the real field $(\mathbb{R}, +, .)$ can describe any function $f : |^{+}_{-}|^p \to |^{+}_{-}|$ (resp. $f : |^{1}_{0}|^p \to |^{1}_{0}|$), where $|^{+}_{-}| = \{-1, +1\} \subset \mathbb{R}$ (resp. $|^{1}_{0}| = \{0, 1\} \subset \mathbb{R}$).*

Proof. Theorem 35 follows from the functional completeness of the nand (or nor) logical operator and the fact that the composition of polynomials in $\mathbb{R}[s_I]$ result in polynomials in $\mathbb{R}[s_I]$. Indeed, the nand operator is defined as a polynomial function with signed (resp. boolean) operands and codomain in Table 5. Thus, the composition of any number of such nand operations on signed (resp. boolean) symbolic variables evaluated in $|^{+}_{-}|$ (resp. $|^{1}_{0}|$) result in a polynomial s-function i.e. a s-polynotope according to the definition 28.

Corollary 36 *Given any pair $(a, b) \in \mathbb{R}^2$ satisfying $a < b$, all the finite dimensional Boolean functions and operations can be plunged in $\{a, b\}^p \subset \mathbb{R}^p$ for some $p \in \mathbb{N}$ after suitable re-scaling compared to the usual Boolean*

[14] Notice that contraposition writes as $(a \leq b) = (\neg b \leq \neg a)$.

case i.e. $(a,b)=(0,1)$. *This is exemplified with the so-called signed case i.e.* $(a,b)=(-1,+1)$ *in table 5, where the direct and inverse affine re-scaling functions are* $r:|^{+}_{-}|\to|^{1}_{0}|, x\mapsto\frac{1+x}{2}$ *and* $r^{-1}:|^{1}_{0}|\to|^{+}_{-}|, x\mapsto 2x-1$.

5.2 Continuous: Nonlinear functions

Whereas the imset (see definition 1) of a polynotope (resp. zonotope) by a polynomial (resp. affine) function is still a polynotope (resp. zonotope), the imset by non-polynomial (resp. non-linear) functions is not. This paragraph proposes a method to obtain guaranteed inclusions of non-polynomial (resp. non-linear) functions while maintaining dependency links between inputs and outputs. Indeed, breaking such links (e.g. by a naive use of interval arithmetic) is often the source of over-approximations and/or wrapping effect.

For the sake of simplified notations, the distinction between s-functions (syntax) and their usual interpretation as a mathematical function (semantic) is not systematic in the following. Example: Let $f(x)=e^x$. Then, f refers either to the s-function built from the wff e^x, or to $\iota_m f : x\mapsto e^x$.
The notations used for intervals are as follows: $x\in[x]=[\underline{x},\overline{x}]=\hat{x}\pm\mathring{x}=[\hat{x}-\mathring{x},\hat{x}+\mathring{x}]$ where $\underline{x}$, $\overline{x}$, $\hat{x}$, $\mathring{x}$ respectively denote the lower bound, upper bound, center (or middle), radius of the interval $[x]$ containing x. Then, $\hat{x}=\frac{\overline{x}+\underline{x}}{2}$ and $\mathring{x}=\frac{\overline{x}-\underline{x}}{2}$. Recall: $\Box=[-1,+1]$.

Lemma 37 (Unit range mapping) *Let* $[x]=[\underline{x},\overline{x}]=\hat{x}\pm\mathring{x}\subset\mathbb{R}$. *Let* $\mu:[x]\to\Box$, $x\mapsto\delta=\frac{x-\hat{x}}{\mathring{x}}$ *if* $\mathring{x}\neq 0$, $\delta=0$ *otherwise.* $\mu^{-1}(\delta)=\hat{x}+\mathring{x}\delta$. *Unless* $\mathring{x}=0$ *(degenerate point case), the unit range mapping* μ *(or* $\mu_{[x]}$*) of* $[x]$ *is linear and bijective: It maps the interval range* $[x]$ *of* x *to the unit interval* $\Box$ *containing any* $\delta=\mu(x)$, $x\in[x]$.

Given an interval $[x]\subset\mathbb{R}$, let $f:[x]\to\mathbb{R}$, $x\mapsto y=f(x)$ be a function that does not satisfy a property π (i.e. $\neg\pi(f)$ is true) required for a given class of image-sets to be closed under the (element-wise) application of f to the underlying s-functions. For example: π being the property of being linear (resp. polynomial), the class of e-zonotopes (resp. e-polytopes) are closed under the application of linear (resp. polynomial) functions to the underlying s-zonotopes (resp. s-polytopes). $\neg\pi(f)$ then means that f is non-linear (resp. non-polynomial) for zonotopes (resp. polynotopes). The considered structural property π is assumed to be preserved through function composition.

Lemma 38 (Generic inclusion method) *Given an interval* $[x]\subset\mathbb{R}$ *and* $f:[x]\to\mathbb{R}$, $x\mapsto y=f(x)$ *with* $\neg\pi(f)$. *Let* $g:\Box^2\to\mathbb{R}$, $(\delta,\epsilon)\mapsto g(\delta,\epsilon)$ *be a function satisfying* $\forall x\in[x], \exists\epsilon\in\Box, f(x)=g(\mu(x),\epsilon)$, *where* μ *is the unit range mapping of* $[x]$. *Then,* $\tilde{f}(.)=g(\mu(.),\Box)$ *is an inclusion function for* $f(.)$. *Also,* $\pi(g)\wedge\pi(\mu)\Rightarrow\pi(\tilde{f})$.

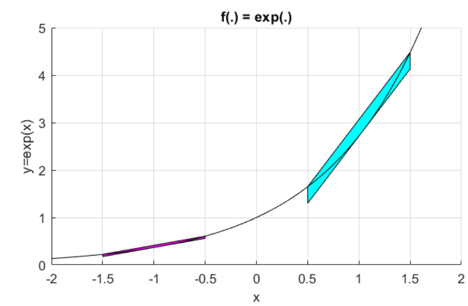

Fig. 3. Inclusion method of theorem 39 applied to $f(x)=e^x$ on $[x]=-1\pm0.5$ (magenta) and $[x]=+1\pm0.5$ (cyan): plot of e-zonotopes as in corollary 40. $\hat{x}+\mathring{x}\delta^*=\log(\mathring{y}/\mathring{x})$.

According to Lemma 38, enclosing a non-linear (resp. non-polynomial) function f in a linear (resp. polynomial) framework can be achieved by finding an adequate linear (resp. polynomial) function g. In order to exemplify the generic inclusion method, a more focused approach is proposed for increasing/decreasing and convex/concave functions f on some interval $[x]$. Notation: $[\partial_x f](.)=\left.\frac{\partial f(x)}{\partial x}\right|_{x=.}$.

Theorem 39 (An inclusion method) *Let* $f:[x]\to\mathbb{R}$, $x\mapsto y=f(x)$ *be a class* $\mathcal{C}^1$ *convex or concave function on a given interval* $[x]=\hat{x}\pm\mathring{x}=[\underline{x},\overline{x}]\subset\mathbb{R}$ *with* $\mathring{x}>0$. *Let* $\underline{y}=f(\underline{x})$, $\overline{y}=f(\overline{x})$, $\hat{y}=(\overline{y}+\underline{y})/2$, $\mathring{y}=(\overline{y}-\underline{y})/2$. *Let* $\delta=\mu(x)$ *where* μ *is the unit range mapping of* $[x]$ *(so,* $x\in[x]\Leftrightarrow\delta\in\Box$*). Let* $r(\delta)=f(\hat{x}+\mathring{x}\delta)-(\hat{y}+\mathring{y}\delta)$. *Let* δ^* *be the solution of* $[\partial_x f](\hat{x}+\mathring{x}\delta)=\mathring{y}/\mathring{x}$. *Then,* $g(\delta,\epsilon)=g_0+g_1\delta+g_2\epsilon$ *with* $g_0=\hat{y}+\frac{1}{2}r(\delta^*)$, $g_1=\mathring{y}$, $g_2=\frac{1}{2}|r(\delta^*)|$ *satisfies* $\forall x\in[x]$, $\exists\epsilon\in\Box$, $f(x)=g(\mu(x),\epsilon)$. $\tilde{f}(.)=g(\mu(.),\Box)$ *is an inclusion function for* $f(.)$ *on* $[x]$.

Proof. The regularity of f on $[x]$ ensures that $[\partial_\delta r](\delta^*)=0$ i.e. $[\partial_x f](\hat{x}+\mathring{x}\delta^*)=\mathring{y}/\mathring{x}$ has a unique solution. Since $r(-1)=r(+1)=0$, if f is convex (resp. concave) on $[x]$, then $r(\delta)\in[r(\delta^*),0]$ (resp. $r(\delta)\in[0,r(\delta^*)]$) for $\delta\in\Box$. The two cases are gathered as: $r(\delta)\in\frac{1}{2}r(\delta^*)\pm\frac{1}{2}|r(\delta^*)|$ Thus, $\exists\epsilon\in\Box$, $r(\delta)=f(x)-(\hat{y}+\mathring{y}\delta)=\frac{1}{2}r(\delta^*)+\frac{1}{2}|r(\delta^*)|\epsilon$, as $x=\hat{x}+\mathring{x}\delta$ by definition of μ as in Lemma 37. Then, $g(\delta,\epsilon)$ and the proof follow from the last equality.

Corollary 40 *The s-function* $[\mu^{-1}(\delta);g(\delta,\epsilon)]$ *where* δ *and* ϵ *refer to symbols of type (unit) interval is a continuous s-zonotope since* μ^{-1} *and* g *are affine.* $\forall x\in[x]$, $[x;f(x)]\in\langle[\mu^{-1}(\delta);g(\delta,\epsilon)]\rangle_{s,\tau,\iota}$, *an e-zonotope usually not reduced to an aligned box due to the dependency of both dimensions on common symbol(s) referred as* δ. *Moreover, if* x *is a polynotope, so is* $\langle[\mu^{-1}(\delta);g(\delta,\epsilon)]\rangle_{s,\tau}$.

An illustrative example with $f(x)=e^x$ is reported in Fig. 3 and further remarks are reported hereafter:
a) The inclusion proposed in theorem 39 is entirely parameterized by the input domain $[x]$ while not being sub-

Table 6
Switching functions expressed from the absolute value operator: $\mathrm{abs}(x) = |x|$ (or from ReLU*).

Function	Notation	Expression with $\lvert.\rvert$ (or pos)
Maximum	$\max(x, y)$	$= \frac{x+y}{2} + \frac{\lvert x-y\rvert}{2}$ $(= y + \mathrm{pos}(x-y))$
Minimum	$\min(x, y)$	$= \frac{x+y}{2} - \frac{\lvert x-y\rvert}{2}$ $(= x - \mathrm{pos}(x-y))$
Saturation	$\mathrm{sat}(x, \underline{x}, \overline{x})$	$= \frac{1}{2}(\underline{x} + \overline{x} + \lvert\underline{x} - x\rvert - \lvert x - \overline{x}\rvert)$
Deadzone	$\mathrm{dz}(x, \underline{x}, \overline{x})$	$= x - \mathrm{sat}(x, \underline{x}, \overline{x})$
ReLU*	$\mathrm{pos}(x)$	$= \max(0, x) = \frac{x+\lvert x\rvert}{2}$

*Rectifier Linear Unit (remark: $|x| = 2\mathrm{pos}(x) - x$).

ject to the arbitrary choice of a point used as reference for linearizing or computing a Taylor expansion.
b) δ^* often has an explicit form, e.g., $\hat{x} + \mathring{x}\delta^* = \log(\mathring{y}/\mathring{x})$, $\mathring{x}/\mathring{y}$, $\frac{1}{4}(\mathring{x}/\mathring{y})^2$ for $f(x) = e^x$, $\log(x)$, $\sqrt{x}$, respectively.
c) If $\mathring{x} = 0$, then the input is a point value and $f(x) = f(\hat{x})$. This is consistent with the limit $\mathring{x} \to 0$ since the continuity of f gives $\mathring{y} \to 0$, $g_0 \to \hat{y}$, $g_1 \to 0$ and $g_2 \to 0$.
d) If f is decreasing, then $\overline{y} < \underline{y}$ and $\mathring{y} < 0$ in theorem 39.
e) $r(\delta)$ is the remainder term wrt to a (linear) approximation of $f(x)$ which itself (linearly) depends on x: $\hat{y} + \mathring{y}\mu(x)$. The purpose of a dependency-preserving inclusion (dpi) is thus achieved, at least for a structural property π referring to being linear. Note that polynomial dependencies possibly modeling x (then, $x(.)$ and $\delta(.) = \mu(x(.))$ are polynomials) are readily propagated by a linear enclosing approximation $\tilde{f}(.)$ of $f(.)$. Indeed, the composition of affine and polynomial functions is still polynomial. Thus, the result in theorem 39 is readily applicable with polynotopes. Moreover, the generic inclusion method in Lemma 38 encompasses polynomial enclosing approximations of non-polynomial functions.

5.3 Hybrid: Switching functions

In the last paragraph (§5.2), an inclusion method for functions f satisfying the regularity conditions of being $\mathcal{C}^1$ has been proposed in Theorem 39. Following the generic inclusion method stated in Lemma 38, the case of a prototypical $\mathcal{C}^0$ but not $\mathcal{C}^1$ function is considered in this paragraph: the absolute value. The motivation for this is summarized in Table 6 which shows that several useful switching functions can be built by composing basic operators (like $+$, $-$, taking the half) with the absolute value operator $abs(x) = |x|$. Thus, a dependency-preserving inclusion of a prototypical switching function like *abs* is highly desirable to model and efficiently propagate uncertainties within hybrid dynamical systems, without necessarily requiring costly bisections and/or a specific management of guard conditions.

Theorem 41 (An inclusion of abs) *Let abs* : $[x] \to \mathbb{R}$, $x \mapsto y = |x|$ *be the restriction of the absolute value on a given interval* $[x] = \hat{x} \pm \mathring{x} = [\underline{x}, \overline{x}] \subset \mathbb{R}$.
Case 1: If $\overline{x} \leq 0$, *then* $abs(.) = -(.)$,
Case 2: If $\underline{x} \geq 0$, *then* $abs(.) = +(.)$,
Case 3: If $|\hat{x}| < \mathring{x}$, *then* $\widetilde{abs}(.)$ *is an inclusion function for* $abs(.)$ *on* $[x]$ *with:*

$$\widetilde{abs}(.) = \left(\frac{\hat{x}}{\mathring{x}}\right)(.) + \left(\frac{\mathring{x}^2 - \hat{x}^2}{2\mathring{x}}\right)(1 + \Box). \tag{19}$$

Proof. If $0 \notin [x]$ (case 1 or 2), $abs(.)$ is linear and no dedicated inclusion is then required. If $0 \in [x]$ (case 3), the inclusion method of theorem 39 is applied step-by-step: Let μ be the unit range mapping of $[x]$ and $\delta = \mu(x)$. Since $abs(.)$ is only $\mathcal{C}^0$ and convex on $[x]$, but not $\mathcal{C}^1$, the range of the remainder $r(\delta) \leq 0$ (still such that $r(-1) = r(+1) = 0$) is computed by noticing that its minimum is obtained for $x = \hat{x} + \mathring{x}\delta^* = 0$. Then, $\delta^* = -\hat{x}/\mathring{x}$ gives $r(\delta^*) = (\hat{x}^2 - \mathring{x}^2)/\mathring{x}$ and satisfies $\forall \delta \in \Box, r(\delta) \in [r(\delta^*), 0]$. It comes $g(\delta, \epsilon) = g_0 + g_1\delta + g_2\epsilon$ with $g_0 = \frac{\mathring{x}^2 + \hat{x}^2}{2\mathring{x}}$, $g_1 = \hat{x}$, $g_2 = \frac{\mathring{x}^2 - \hat{x}^2}{2\mathring{x}}$ ($|\hat{x}| < \mathring{x}$ in case 3). Finally, $g(\mu(x), \epsilon) = (\frac{\hat{x}}{\mathring{x}})x + (\frac{\mathring{x}^2 - \hat{x}^2}{2\mathring{x}})(1 + \epsilon)$.

Corollary 40 still applies to $f = abs$, as a corollary of theorem 41 rather than theorem 39. A dependency-preserving inclusion of $abs(.)$ has been obtained. By extension, dependency-preserving inclusions (dpi) for the switching functions reported in table 6, among others possibly resulting from functional compositions are also obtained. Moreover, the vertical concatenation operator implemented for zonotopes and polynotopes allows to build n-dimensional dpi from scalar ones through basic compositions. These can be implemented by using the overloading capability of some object oriented languages, to the benefit of code readability. This feature holds not only for switching functions, but also for non-linear/non-polynomial ones. This makes polynotopes a relevant tool to compute and analyze mixed uncertainty propagation within non-linear hybrid dynamical systems. Indeed, their polynomial nature, efficiently encoded by combining full and sparse data structures, looks appropriate to model a wide spectrum of non-trivial dependencies, as shown by the functional completeness result given in theorem 35.

6 Polynotopic Kalman Filter (PKF)

An extension of Kalman Filtering to discrete-time non-linear hybrid dynamical systems is proposed in this section. It is based on polynotopes and interpretations related to a set-membership uncertainty paradigm.

Let $x(s)$ be a s-polynotope (8): $x(s) = \langle c, R, I, E\rangle_{s,\tau} = c + Rs_I^E$ (syntax). By analogy with zonotopes, its covariation [6] is defined as: $\mathrm{cov}(x(s)) = RR^T$. In order to possibly take symbol types and/or the monomial structure into account [15], a covariation weighted by Φ (pos-

[15] e.g. to weight the relative influence of continuous and discrete symbols/uncertainties on the accuracy criterion further chosen to optimize the mixed-set based state estimates obtained from PKF.

sibly $\Phi(I,E)$ or any $\Phi(.)$ depending on known values) is introduced as:

Definition 42 (Weighted covariation) *Given a symmetric matrix Φ ($\Phi = \Phi^T$), the weighted covariation of $\mathbf{x} = \langle c, R, \ldots\rangle_{s,\tau(,\iota)}$ (polynotope or zonotope) is:*

$$\mathrm{cov}_\Phi(\mathbf{x}) = R\Phi R^T.$$

$x(s)$ formalizes a vector polynomial (s-)function of the symbolic variables in s_I. The execution of polynotope operations like sum, linear image, concatenation, reduction, zonotopic hull $\mathcal{Z}x(s)$, interval/box hull $\mathcal{B}x(s)$, etc mainly work at a syntactic level by manipulating polynomial expressions (e.g. encoded as (c,R,I,E) with sparse E) while preserving semantic properties. In particular, inclusion is viewed as a semantic property related to a set-membership interpretation of polynomial functions depending on typed [16] symbolic variables. From (9), it comes: $\forall \sigma \in \iota\tau s_I, \iota x(\sigma) \in \mathcal{P}x(s) = \langle c,R,I,E\rangle_{s,\tau,\iota}$ (semantics), where $\iota x(.)$ stands for the interpretation of $x(.)$ as a vector of polynomial mathematical functions ($\mathbb{R}^{dim(I)} \to \mathbb{R}^{n_x}$) with real coefficients (under assumption 6). Since probability theory is the most commonly used framework for nonlinear filtering, some analogies and exploratory links are briefly outlined as a remark:

Remark 43 *A first idea to introduce probability theory in the proposed scheme simply consists in extending the basic symbol types in assumption 6 to other types like (some class of) random variables defined on a given probability space. This can work in the linear case with symbolic zonotopes and/or independent Gaussian random variables (e.g. see §2.2 and definition 6.1 in [9]). Pushing further in such a direction could be an option. Another one may rely on interpreting σ (as in definition 13, (5) and (9)) as an "outcome", the function $\iota x(.)$ as a "random variable", $[\iota x]^{-1}(S)$ as an "event" related to any set S of output values taken by $\iota x(.)$. A measure $\pi(.)$ of events on the domain $\iota\tau s_I$ induced by an interpretation of symbol types would then become some kind of conditional probability wrt $\iota\tau s_I$. The typed symbols s_I would then contribute to define the probability space itself.*

In the following, no probability measure is considered. Notations: $\mathbf{x} = x(s)$ denotes a s-polynotope. $x = \iota x(\sigma) \in \mathbb{R}^{n_x}$ denotes a point evaluation of $\mathbf{x}$ obtained for some so-called outcome $\sigma \in \iota\tau s_I$. Then, $x \in \mathcal{P}\mathbf{x}$, the e-polynotope related to $\mathbf{x}$. Also, $x \in \mathcal{Z}\mathbf{x}$ (resp. $x \in \mathcal{B}\mathbf{x}$) means that x belongs to a zonotopic (resp. interval/box) hull of $\mathbf{x}$.

The state observation (or filtering) problem addressed in this section deals with discrete-time non-linear hybrid dynamical systems modeled as:

$$x_+ = f(x,u,v), \quad x_0 \in \mathcal{P}\mathbf{x}_0,\, v \in \mathcal{P}\mathbf{v}, \tag{20}$$
$$0 = g(x,u,v,y), \tag{21}$$

where the functions $f(.)$ and $g(.)$ result from the composition of elementary functions and operators for which inclusion preserving polynotope versions are available. In practice, this is not much restrictive since sum, linear image, reduction, concatenation, product are available (see §4.3) and the modeling tools for nonlinear hybrid systems developed in section 5 can be used to that purpose. Then, by overloading [17] these elementary functions and operators with their inclusion preserving polynotopic version, and by applying the same composition, inclusion functions with polynotopic inputs and outputs $\tilde{f}(.)$ and $\tilde{g}(.)$ can be obtained for $f(.)$ and $g(.)$, respectively. In (20), the index $+$ refers to the next time step $k+1$ and the current time step k is omitted to simplify the notations, except for the initial state x_0 at time $k=0$. $x_0 \in \mathbb{R}^{n_x}$ is assumed unknown but bounded by the e-polynotope $\mathcal{P}\mathbf{x}_0$ related to a known s-polynotope $\mathbf{x}_0$. $x \in \mathbb{R}^{n_x}$, $u \in \mathbb{R}^{n_u}$, $y \in \mathbb{R}^{n_y}$, $v \in \mathbb{R}^{n_v}$ respectively stand for the states, the known (control) inputs, the known measurements, the unknown but bounded uncertainties (state and measurement noises, disturbances, modeling errors, etc) at time k. v is assumed bounded by a known polynotope $\mathcal{P}\mathbf{v}$. Notice that $u \in \mathcal{P}\mathbf{u} = \{u\}$ (singleton) for $\mathbf{u} = \langle u, \emptyset, \emptyset, \emptyset\rangle_{s,\tau}$. Similarly, $y \in \mathcal{P}\mathbf{y}$ with $\mathbf{y} = \langle y, \emptyset, \emptyset, \emptyset\rangle_{s,\tau}$. The problem addressed is that of designing a one step-ahead prediction filter (or state observer) minimizing the trace tr(.) of the (weighted) covariation of a polynotope enclosing the predicted state.

Filtering is mainly a data fusion process. So, how to merge (vector) sources? Weighting is a usual solution:

$$z_1 \in \mathcal{P}\mathbf{z}_1 \wedge z_2 \in \mathcal{P}\mathbf{z}_2 \Rightarrow z = G_1 z_1 + G_2 z_2 \in \mathcal{P}\mathbf{z} \quad \text{with } \mathbf{z} = G_1\mathbf{z}_1 + G_2\mathbf{z}_2. \tag{22}$$

Two noticeable ways to particularize (22) are:
a) Taking $z_1 = z_2$ under $G_1 + G_2 = \mathcal{I}$ gives (23) which parameterizes enclosures of a polynotope intersection that could be used to design a state bounding observer:

$$z \in (\mathcal{P}\mathbf{z}_1 \cap \mathcal{P}\mathbf{z}_2) \Rightarrow z \in \mathcal{P}(G_1\mathbf{z}_1 + G_2\mathbf{z}_2). \tag{23}$$
$$z \in \mathcal{P}\mathbf{z}_1 \wedge 0 \in \mathcal{P}\mathbf{z}_2 \Rightarrow z \in \mathcal{P}(\mathbf{z}_1 - G\mathbf{z}_2). \tag{24}$$

b) Taking $z_2 = 0$ and $G_1 = \mathcal{I}$ under $G_2 = -G$ gives (24) which parameterizes an update (or correction) of an initial knowledge $\mathcal{P}\mathbf{z}_1$ about z with some other depending knowledge, $\mathcal{P}\mathbf{z}_2$, such as the one obtained through some measurements. (24) thus looks as a prototypical weighting underlying the structure of Kalman Filters. Moreover, in our approach, the symbols possibly shared be-

[16] Notice that the set-membership interpretation is also related to the types considered under the assumption 6.

[17] To the benefit of code readability and maintainability.

Table 7
PKF iteration: $\mathbf{x}_+ = \text{PKF}(\mathbf{x},\mathbf{u},\mathbf{v},\mathbf{y},\tilde{f},\tilde{g},\Phi,q)$:

$$\bar{\mathbf{x}} = \downarrow_q \mathbf{x}, \quad \textit{reduction} \quad (25)$$
$$\mathbf{p} = \tilde{f}(\bar{\mathbf{x}},\mathbf{u},\mathbf{v}), \quad \textit{prediction} \quad (26)$$
$$\mathbf{e} = \tilde{g}(\bar{\mathbf{x}},\mathbf{u},\mathbf{v},\mathbf{y}), \quad \textit{innovation} \quad (27)$$
$$\left\langle \check{c}, \begin{bmatrix} R_p \\ R_e \end{bmatrix}, \check{I}, \check{E} \right\rangle_{s,\tau} = \begin{bmatrix} \mathbf{p} \\ \mathbf{e} \end{bmatrix}, \quad \textit{alignment} \quad (28)$$
$$G = (R_p \Phi R_e^T)(R_e \Phi R_e^T)^{-1}, \quad \textit{optimal gain} \quad (29)$$
$$\mathbf{x}_+ = \mathbf{p} - G\mathbf{e}. \quad \textit{update} \quad (30)$$

tween $\mathbf{z}_1$ and $\mathbf{z}_2$ play a key role in the modeling of dependencies. This makes it possible to tune/optimize G so as to maximize uncertainty cancellation [18] when computing $\mathbf{z}_1 - G\mathbf{z}_2$. The general idea of Kalman Filters is indeed to optimize the precision of a prediction $\mathbf{p} = \mathbf{z}_1$ by using a dependent yet complementary source, the innovation $\mathbf{e} = \mathbf{z}_2$, to update the prediction as $\mathbf{p} - G\mathbf{e}$ (30). Then, the algorithm (25)$-$(30) implementing an iteration of the proposed Polynotopic Kalman Filter (PKF) follows as in Table 7 and theorem 44.

Theorem 44 (PKF: inclusion and optimal gain) *Given a system modeled as in (20)$-$(21), the PKF iteration in (25)$-$(30) (Table 7) satisfies a) and b):*
a) $x \in \mathcal{P}\mathbf{x} \wedge v \in \mathcal{P}\mathbf{v} \Rightarrow x_+ \in \mathcal{P}\mathbf{x}_+$,
b) Let $G^* = \arg\min_G \text{tr}(\text{cov}_\Phi(\mathbf{x}_+))$. *$G^*$ is the optimal gain computed as in (29):* $G^* = (R_p \Phi R_e^T)(R_e \Phi R_e^T)^{-1}$.

Proof. *a)* : By construction, $\tilde{f}(.)$ and $\tilde{g}(.)$ are inclusion functions for $f(.)$ and $g(.)$. Since the reduction step (25) is inclusion preserving, the inclusion property *a)* is a direct consequence of (24) with $\mathbf{z}_1 = \mathbf{p}$ and $\mathbf{z}_2 = \mathbf{e}$.
b) : $\partial_X h(X)$ denoting $\partial h(X)/\partial X$, if $h(.)$ returns scalar values and $X = [X_{ij}]$ is a matrix, then $\partial_X h(X) = [\partial_{X_{ji}} h(X)]$. X, A, B, C being matrices of correct size,

$$\partial_X \text{tr}(AX^TB) = A^TB^T, \quad (31)$$
$$\partial_X \text{tr}(AXBX^TC) = BX^TCA + B^TX^TA^TC^T. \quad (32)$$

Let $J(G) = \text{tr}(\text{cov}_\Phi(\mathbf{x}_+))$. In (28), $\check{c} = [c_p; c_e]$ and $[\mathbf{p};\mathbf{e}]$ is such that [19] $\mathbf{p} = c_p + R_p s_{\check{I}}^{\check{E}}$ and $\mathbf{e} = c_e + R_e s_{\check{I}}^{\check{E}}$. From (30), $\mathbf{x}_+ = \langle c_p - Gc_e, R_p - GR_e, \check{I}, \check{E}\rangle_{s,\tau}$ and $J(G) = \text{tr}((R_p - GR_e)\Phi(R_p - GR_e)^T) = \text{tr}(R_p\Phi R_p^T) - 2\text{tr}(R_p \Phi R_e^T G^T) + \text{tr}(GR_e\Phi R_e^T G^T)$. Using (31) and (32), $\partial_G J(G) = -2(R_e \Phi R_p^T) + 2(R_e\Phi R_e^T)G^T$. G^* being the value of G such that $\partial_G J(G) = 0$, it comes $G^* R_e \Phi R_e^T = R_p \Phi R_e^T$ and $G^* = (R_p\Phi R_e^T)(R_e\Phi R_e^T)^{-1}$.

[18] which is impossible with usual interval arithmetic, subject to the so-called dependency problem.

[19] The polynotope concatenation $[\mathbf{p};\mathbf{e}]$ gives expressions of $\mathbf{p}$ and $\mathbf{e}$ such that the generators related to their common monomials (i.e. dependencies) become "aligned" in the same columns of the matrices R_p and R_e. This is the reason why (28) is called the alignment step.

Theorem 45 (PKF vs. ZKF) *Let consider the particular case of linear functions $f(.)$ and $g(.)$ defined as:*
$f(x,u,v) = Ax + Bu + Ev_p$,
$g(x,u,v,y) = Cx + Du + Fv_e - y$,
where $v = [v_p; v_e]$ (state noise and measurement noise), and A, B, C, D, E, F are (possibly time-varying) matrices with appropriate dimensions. Only symbols of type (unit) interval are considered and $\Phi = \mathcal{I}$. Also, let $\tilde{f}(.) = f(.)$, $\tilde{g}(.) = g(.)$, $x_0 \in \mathcal{P}\mathbf{x}_0$ with $\mathbf{x}_0 = \langle c_0, R_0, I_0, \mathcal{I}\rangle_{s,\tau}$ (then, $\mathcal{P}\mathbf{x}_0 = \mathcal{Z}\mathbf{x}_0$ is a zonotope), $v \in \mathcal{P}\mathbf{v}$ with $\mathbf{v} = \langle 0, \mathcal{I}, I_v, \mathcal{I}\rangle_{s,\tau}$ (then, $\mathcal{P}\mathbf{v} = \mathcal{Z}\mathbf{v} = \mathcal{B}\mathbf{v}$ is a unit hypercube). It is also assumed that I_0 and all I_v's have no common scalar elements/identifiers which are all unique (then, no symbol being shared between $\mathbf{x}_0$ and all the $\mathbf{v}$'s, this is in fact an independence assumption). Then, PKF computes the same centers c (state point estimates) and generator/shape matrices R as ZKF in [6] would do, up to column permutations; all the computed polynotopes are also zonotopes, and the optimal gain $G = AK$ corresponds to the usual Kalman gain $K = \bar{P}C^T(C\bar{P}C^T + FF^T)^{-1}$ with $\bar{P} = \bar{R}\bar{R}^T$.

Proof. Polynotopes (and zonotopes) being closed under linear transforms, taking $\tilde{f}(.) = f(.)$ and $\tilde{g}(.) = g(.)$ suffices to preserve inclusion when (20)$-$(21) is a discrete-time LTV (or LTI) model. Moreover, all the polynotope exponent matrices equaling $\mathcal{I}$, (s-)polynotope operations naturally reduce to (s-)zonotope operations, and the generator matrix computed by the considered reduction operator does not depend on the symbolic description. The focus of the proof is first placed on the observer structure and, then, on the optimal gain. Observer structure:
(25): $\bar{\mathbf{x}} = \langle \bar{c}, \bar{R}, \bar{I}, \mathcal{I}\rangle_{s,\tau} = \downarrow_q \mathbf{x}$ where $\bar{c} = c$,
(26): $\mathbf{p} = A\bar{\mathbf{x}} + B\mathbf{u} + E\mathbf{v_p}$, with $\mathbf{v_p} = [\mathcal{I}, 0]\mathbf{v}$,
(27): $\mathbf{e} = C\bar{\mathbf{x}} + D\mathbf{u} + F\mathbf{v_e} - \mathbf{y}$, with $\mathbf{v_e} = [0, \mathcal{I}]\mathbf{v}$,
(30): $\mathbf{x}_+ = \mathbf{p} - G\mathbf{e}$ gives:
$\mathbf{x}_+ = (A\bar{\mathbf{x}} + B\mathbf{u} + E\mathbf{v_p}) + G(\mathbf{y} - (C\bar{\mathbf{x}} + D\mathbf{u} + F\mathbf{v_e}))$,
which corresponds to ⟨(14)⟩ i.e. the equation (14) in [6] where (v, w) stands for (v_p, v_e). Also, just for insight:
$\mathbf{x}_+ = (A - GC)\bar{\mathbf{x}} + (B - GD)\mathbf{u} + [E, -GF]\mathbf{v} + G\mathbf{y}$.
Keeping in mind that the sum of two generators with the same monomial term (here: with the same symbol) is a classical vector sum, and an horizontal concatenation otherwise (see MLC in §4.1), the centers c_* and generator matrices R_* of the s-polynotopes (also s-zonotopes since $E_* = \mathcal{I}$) computed in (26), (27) and (30) are [20]:
(26): $c_p = A\bar{c} + Bu$,
(26): $R_p = [A\bar{R}, E]$ since $\bar{I} \cap I_{v_p} = \emptyset$,
(27): $c_e = C\bar{c} + Du - y$,

[20] Up to column permutations with no impact on the interpretation.

Table 8
Algorithm of functions building Half (H), Full 1 bit (F), and Full n bits (N) adder with *nand* gates ($\neg x \leftrightarrow x \barwedge x$).

$H : (a, b) \mapsto (s, c)$:	Half-adder with 5 nand gates
$t_1 \leftarrow a \barwedge b$, $t_2 \leftarrow a \barwedge t_1$, $t_3 \leftarrow t_1 \barwedge b$, $s \leftarrow t_2 \barwedge t_3$, $c \leftarrow t_1 \barwedge t_1$.	
$F : (a, b, c_{in}) \mapsto (s, c_{out})$:	Full 1 bit adder with carry
$(r, c_1) \leftarrow H(a, b)$, $(s, c_2) \leftarrow H(r, c_{in})$, $c_{out} \leftarrow \neg c_1 \barwedge \neg c_2$.	
$N : (A, B, c) \mapsto (S, c)$:	Full n bits adder with carry
for $i \leftarrow 1 \dots n(A)$, $(S(i), c) \leftarrow F(A(i), B(i), c)$.	

(27): $R_e = [C\bar{R}, F]$ since $\bar{I} \cap I_{v_e} = \emptyset$,
(30): $c_+ = c_p - Gc_e = (A - GC)\bar{c} + (B - GD)u + Gy$,
(30): $R_+ = R_p - GR_e = [(A - GC)\bar{R}, E, -GF]$,
since $I_{v_p} \cap I_{v_e} = \emptyset$, but note that PKF can take dependent state and measurement noises into account with $\mathbf{v}$. Finally, it can be checked that c_+ and R_+ exactly coincide with ((15)) and ((16)), respectively, so proving that PKF reduces to the same observer structure as ZKF under the specific assumptions of theorem 45.
Optimal gain: Let $\bar{P} = \bar{R}\bar{R}^T$ ($= \text{cov}_{\Phi}(\bar{\mathbf{x}})$, $\Phi = \mathcal{I}$). Respectively substituting $[A\bar{R}, E, 0]$ and $[C\bar{R}, 0, F]$ for R_p and R_e in (29) gives $G = AK$ with $K = \bar{P}C^T(C\bar{P}C^T + FF^T)^{-1}$. Then, it can be checked that the optimal observer gain is the same as in ((21)) $-$ ((22)).

Remark 46 (PKF vs. KF) *Theorem 45 (PKF vs. ZKF) can be combined with Theorem 7 (ZKF vs. KF) in [6] to make a further bridge between set-membership and stochastic paradigms. In particular, this gives the conditions under which PKF covariations and KF covariances coincide, as well as the state point estimates.*

Based on modeling tools for nonlinear hybrid systems developed in the proposed approach, a compositional implementation of advanced reachability and filtering algorithms preserving inclusion is made possible by using operator overloading. This is exemplified with the Polynotopic Kalman Filter (PKF) proposed in this section.

7 Numerical Examples

7.1 Discrete: Adder

The first example illustrates some connection with basic digital circuit design. The s-polynotopes (i.e. polynomial s-functions) resulting from the multi-affine decomposition of n bits binary adders only made of *nand* gates are compared depending on the type of symbol(ic variable)s used: signed or boolean as explained in §5.1.

The binary adders architecture is described in Table 8 where $S(i)$ refers to the projection of the s-polynotope S along the ith dimension, $i = 1, \dots, n(S)$. For an n bits adder, $A = s_{1:n}$ and $B = s_{(n+1):2n}$ each refer to a vector of n (either signed or boolean) symbolic variables representing the (unknown) bits encoding two integer operands. The e-zonotope (or e-polynotope) related to A is the set of the 2^n possible input values i.e. $\{-1, +1\}^n$ (resp. $\{0, 1\}^n$) in the signed (resp. boolean) case. Idem for B. These sets are never computed explicitly: they only describe the set-valued interpretation of semi-symbolic calculi based on the (c, R, I, E) data structure used to represent s-polynotope objects. Then, building the architecture of an n bits adder by computing $(S, c) = N(A, B)$ as in table 8 with s-polynotope overloaded operators results in the s-polynotopes S (sum result) and c (output carry) of dimension n and 1, respectively. $S(i)$ is a polynomial with scalar coefficients giving the expression of the ith bit of S as a function of the input bits/symbols in A, B and an input carry. A full n-bits adder has $2n + 1$ (binary) inputs and $n + 1$ (binary) outputs. The s-polynotope $\tilde{S} = [S; c]$ gathering the sum result and the output carry is thus given by $\tilde{S} = \langle \tilde{c}, \tilde{R}, \tilde{I}, \tilde{E} \rangle_{s,\tau}$ with $\tilde{c} \in \mathbb{R}^{n+1}$, $\tilde{R} \in \mathbb{R}^{(n+1)\times m(n)}$, $\tilde{I} \in \mathbb{N}^{2n+1}$, $\tilde{E} \in \mathbb{N}^{(2n+1)\times m(n)}$. The number of (distinct) generators/monomials in $\tilde{S}$, including the center/constant term, is $1 + m(n)$. This number is reported in Table 9 depending on the number n of bits of the adder and the symbol types: either signed or boolean. An unexpected yet interesting result is obtained: The number of distinct monomials required to describe the full architecture of an n-bits adder is much smaller and more scalable using signed rather than boolean symbols.

Table 9
Number of distinct generators/monomials of s-polynotopes representing the multi-affine decomposition of an n-bits adder (as in table 8) with either signed or boolean symbol(ic variable)s. The computation time is given in seconds (s).

n	1	2	3	4	5	6	7	8
Signed	5	11	23	47	95	191	383	767
(s)	0.01	0.02	0.03	0.06	0.15	0.36	1.9	8.3
Boolean	8	23	65	188	554	1649	-	-
(s)	0.01	0.01	0.03	0.2	2.3	29	-	-

This example of a full n-bits adder shows the ability of s-polynotopes to describe and manipulate purely discrete expressions yielding non trivial relations and dependencies between inputs and outputs. Though requiring further studies, the automatic reduction if such relations could help to struggle against combinatorial explosion by gathering into continuous domains the influence of many bits/signs having a small influence on a given criterion, while keeping trace of how the most significant ones influence that criterion. Moreover, the polynomial representation benefits from useful simplifications made possible by dealing with typed symbols.

7.2 Continuous: Van-Der-Pol oscillator

In order to illustrate reachability on continuous domains and compare the results with [22], a Van-Der-Pol oscil-

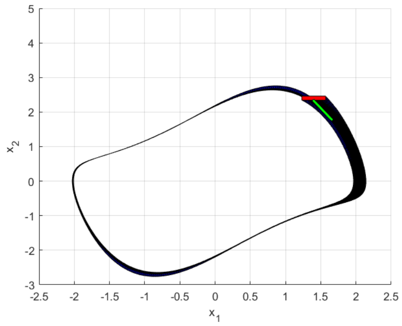

Fig. 4. Reachability result obtained on the Van-Der-Pol oscillator with continuous polynotopes (the plot results from zonotopic enclosures $\mathcal{Z}\mathbf{x}$ at each time step).

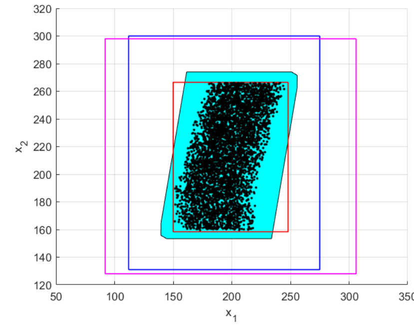

Fig. 5. Reachability result obtained with polynotopes (cyan) for the model of a 3-link traffic network representing a *diverge* junction. Comparison with the results reported in Figure 2 in [28]: methods C/GB (blue), SDMM-IA (magenta), MM and SDMM-S/F (red); Monte-Carlo simulations (black dots).

lator taken from [17] is considered:

$$\begin{aligned}\dot{x}_1 &= x_2,\\ \dot{x}_2 &= (1-x_1^2)x_2 - x_1.\end{aligned}$$

The initial state set is $\mathcal{P}\mathbf{x}_0 = \mathcal{P}[\mathbf{x}_{1,0}; \mathbf{x}_{2,0}] = [[1.23, 1.57]; [2.34, 2.46]]$ as shown by the red box in Fig. 4. 1360 iterations based on an Euler sampling with step size $h = 0.005$ are computed in 9.8 s with continuous polynotopes under Matlab running on a 1.8 GHz Core i5 processor with 8 Go RAM. The zonotopic enclosure $\mathcal{Z}\mathbf{x}_{1360}$ of the final polynotope (at $t = 6.8$ s) is the green set in Fig. 4. At each iteration, the (polynotopic version of the) reduction operator $\downarrow_q$ from [9] with $q = 50$ is used to: *a*) reduce the square $\mathbf{x}_1^2$, *b*) reduce the product $(\mathbf{x}_1^2)\mathbf{x}_2$, *c*) reduce $\mathbf{x}$. As expected, the reachability result shown in Fig. 4 is close to the one obtained with sparse polynomial zonotopes (spz) in the Figure 6 of [22], where a comparison with other methods is conducted. Thus, continuous polynotopes also outperform zonotopes and quadratic zonotopes on this example, which illustrates the interest in dealing with polynomial dependencies to propagate continuous domains within nonlinear dynamics.

7.3 Hybrid: Traffic network

In order to illustrate reachability for dynamics defined with switching functions like min (see §5.3 and table 6) and compare the results with those reported for the TIRA toolbox in [28], the model of a 3-link traffic network representing a *diverge* junction is considered:

$$\begin{aligned}\dot{x}_1 &= -k(x)/T + p,\\ \dot{x}_2 &= k(x)/2 - \min(c, vx_2),\\ \dot{x}_3 &= k(x)/2 - \min(c, vx_3),\\ \text{where } k(x) &= \min(c, vx_1, 2w(\bar{x}-x_2), 2w(\bar{x}-x_3)).\end{aligned}$$

$\min(.,.,.,.)$ is implemented as $\min(\min(.,.), \min(.,.))$. The state $x \in \mathbb{R}^3$ is the vehicle density on each link. $p \in [4/3, 2]$ is the constant but uncertain vehicle inflow to link 1. Notice that the constant nature of this uncertainty is naturally handled by the proposed symbolic approach (no new symbol at each time for $\mathbf{p}$). As in [28], the known parameters of the network are $[T, c, v, \bar{x}, w] = [30, 40, 0.5, 320, 1/6]$. The initial state set is $\mathcal{P}\mathbf{x}_0 = [[150, 200]; [180, 300]; [100, 220]]$. An Euler sampling with step size $h = 1$ and final time $t_f = 30$ is considered. The reduction $\mathbf{x} = \downarrow_q \mathbf{x}$ is applied at each iteration with $q = 20$. Then, 0.26 s were required to compute the polynotope $\mathcal{P}\mathbf{x}_{t_f}$ reported in cyan in Fig. 5. For the sake of a first comparison, the results in the Figure 2 in [28] are also reported in Fig. 5 : C/GB (Contraction/-Growth Bound), MM (Mixed Monotonicity), SDMM-IA (Sampled Data MM-Interval arithmetic), SDMM-S/F(Sampled Data MM-Sampling/Falsification). $\mathcal{P}\mathbf{x}_{t_f}$ looks competitive wrt to the results obtained with TIRA (red box in Fig. 5). Moreover, the computed polynotope captures the orientation of the "black cloud of sample successors" obtained from 5000 Monte-Carlo simulations and also reported in Fig. 5. This illustrates the ability of the proposed scheme to maintain dependency links while propagating uncertainties through hybrid dynamics modeled with switching functions.

7.4 Reachability and Filtering: Lotka-Volterra

A non-linear non-autonomous prey-predator model resulting from the discretization of a modified continuous-time Lotka-Volterra model illustrates *a*) the computation of reachable sets based on a mixed-encoding (§4.2) of the initial state set, and *b*) Polynotopic Kalman Filtering (PKF) as developed in section §6. The modified continuous-time Lotka-Volterra model is $\dot{x} = \mathsf{f}(x, u)$ with $x \in \mathbb{R}^2$, $u \in \mathbb{R}$, $(a, b, c, d) = (2, 0.4, 1, 0.1)$, and $\mathsf{f}(x, u) = [ax_1 - bx_1x_2;\ -cx_2 + dx_1x_2 + u]$.

7.4.1 Reachability with a mixed-encoding of state sets

A mixed-encoding of the initial state set is first considered and further propagated using mixed polynotope

computations within the non-linear dynamics of the discretized Lotka-Volterra model.
More precisely, following the definition 23 and the corollary 24, a 3-level signed-interval mixed encoding of the interval 15 ± 1 is taken as polynotopic initial state set for both $x_{1,0}$ and $x_{2,0}$ (i.e. x_1 and x_2 at $t = kh = 0$):

$$\mathbf{x}_{1,0} = 15 + 1.Z_s^3([!(3,\mathtt{s});!(1,\mathtt{i})]), \tag{33}$$
$$\mathbf{x}_{2,0} = 15 + 1.Z_s^3([!(3,\mathtt{s});!(1,\mathtt{i})]), \tag{34}$$
$$\mathcal{P}\mathbf{x}_0 = \mathcal{P}[\mathbf{x}_{1,0};\mathbf{x}_{2,0}] = [[14,16];[14,16]]. \tag{35}$$

Each occurrence of $!(3,\mathtt{s})$ (resp. $!(1,\mathtt{i})$) calls USP (see §3.3) which returns 3 (resp. 1) unique identifiers of symbols of type signed (resp. unit interval). Thus, $\mathbf{x}_{1,0}$ and $\mathbf{x}_{2,0}$ are independent since they share no common symbol. Note that the symbol types are compatible with the definition 23 of Z_s^g. g refers to the granularity level of the mixed-encoding. $g = 3$ means that 3 signed symbols are used to hierarchically decompose the range 15 ± 1 into $2^3 = 8$ sub-intervals. The coverage of the continuous domain 15 ± 1 is then ensured by the remainder term modeled by the symbol of type unit interval uniquely identified by $!(1,\mathtt{i})$. For instance, let $I = [!(3,\mathtt{s});!(1,\mathtt{i})]$ and $\mathbf{x}_{1,0} = 15 + 1.Z_s^3(I)$ as in (33). Then, $\mathbf{x}_{1,0} = \langle 15, [\frac{1}{2},\frac{1}{4},\frac{1}{8},\frac{1}{8}], I, 1\rangle_{s,\tau} = \frac{1}{2}s_{I_1} + \frac{1}{4}s_{I_2} + \frac{1}{8}s_{I_3} + \frac{1}{8}s_{I_4}$, where the 3 symbols $s_{I_{1:3}}$ are of type signed i.e. $\iota s_{I_{1:3}} \in \{-1,+1\}^3$ and the symbol s_{I_4} is of type (unit) interval i.e. $\iota s_{I_4} \in [-1,+1]$, so that $\mathbf{x}_{1,0}$ also writes as:

$$\mathbf{x}_{1,0} = \frac{1}{2}\boxed{\pm}|_{I_1} + \frac{1}{4}\boxed{\pm}|_{I_2} + \frac{1}{8}\boxed{\pm}|_{I_3} + \frac{1}{8}\square_{I_4}. \tag{36}$$
$$\mathbf{x}_{2,0} = \frac{1}{2}\boxed{\pm}|_{J_1} + \frac{1}{4}\boxed{\pm}|_{J_2} + \frac{1}{8}\boxed{\pm}|_{J_3} + \frac{1}{8}\square_{J_4}. \tag{37}$$

$\mathbf{x}_{2,0}$ is obtained analogously from $J = [!(3,\mathtt{s});!(1,\mathtt{i})]$ ($I \cap J = \emptyset$) and $\mathbf{x}_{2,0} = 15 + 1.Z_s^3(I)$ as in (34). The e-polynotope (or e-zonotope) related to the s-polynotope $\mathbf{x}_{1,0}$ (or s-zonotope since $E = \mathcal{I}$) is thus $\mathcal{P}\mathbf{x}_{1,0} = \mathcal{Z}\mathbf{x}_{1,0} = 15 \pm 1$ (corollary 25). The independence of $\mathbf{x}_{1,0}$ and $\mathbf{x}_{2,0}$ coming from $I \cap J = \emptyset$ gives (35). The polynotope $\mathbf{x}_0 = [\mathbf{x}_{1,0};\mathbf{x}_{2,0}]$ contains all the information required to decompose the initial state set $[[14,16];[14,16]]$ into a paving as in Fig. 6 for $k = 0$. Moreover, assigning values $+1$ or -1 to evaluate some of (or all) the signed symbols in (36)−(37) makes it possible to query about the range covered under some conditional values of signed symbols. This feature makes it possible to trace how each cell and/or cell groups within the “implicit paving” of the initial state set will propagate. It is worth underlining that this can be achieved *without any bissection*, only through the polynomial computations implementing the basic polynotope operators.

For the sake of illustration, an Euler sampling of the Lotka-Volterra model is considered: $x_+ = x + \mathsf{f}(x,0)h$, where the time index k is omitted, x_+ stands for x_{k+1},

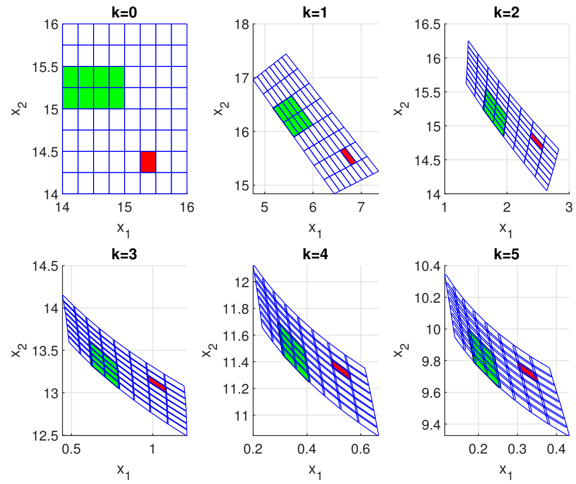

Fig. 6. Reachable sets resulting from a 3-level signed-interval mixed encoding of the initial states of a Lotka-Volterra model. The mixed polynotope computed at each time k characterizes (overlapping) outer approximations of the propagation of each of the 64 cells (red) paving the initial state set *with no bissection*. This also works with cell groups (green).

and the step size is $h = 0.15\,\mathrm{s}$. The reduction operator $\downarrow_{50}$ is applied at each iteration. The reachability analysis reported in Fig. 6 results from $N = 5$ iterations starting from the initial state (33)−(34) satisfying (35). Let $[+-+]$ be a short notation for $[+1;-1;+1]$ (also applying for other sign combinations). Let $\mathbf{x}|(s_I = v)$ denote the s-polynotope obtained by substituting in $\mathbf{x}$ the expressions in v for the symbolic variables in s_I. Unless v depends on some symbols in s_I, $\mathbf{x}|(s_I = v)$ does not depend anymore on s_I. The related e-polynotope (resp. e-zonotopic enclosure) is $\mathcal{P}\mathbf{x}|(s_I = v)$ (resp. $\mathcal{Z}\mathbf{x}|(s_I = v)$). Considering the vectors of unique symbol identifiers I and J as in (36)−(37), the red (resp. green) cell at $k = 0$ in Fig. 6 corresponds to $\mathcal{P}\mathbf{x}_0|(s_{I_{1:3}} = [+-+], s_{J_{1:3}} = [--+])$ (resp. $\mathcal{P}\mathbf{x}_0|(s_{I_1} = [-], s_{J_{1:2}} = [+-])$). Then, the single s-polynotope $\mathbf{x}_k$ computed at each iteration k contains all the information required to obtain the related subplot in Fig. 6. In particular, the red (resp. green) zonotopic sets are $\mathcal{Z}\mathbf{x}_k|(s_{I_{1:3}} = [+-+], s_{J_{1:3}} = [--+])$ (resp. $\mathcal{Z}\mathbf{x}_k|(s_{I_1} = [-], s_{J_{1:2}} = [+-])$) for $k = 1,\ldots,5$. These zonotopic enclosures are guaranteed to enclose the set of states reached from the initial red cell (resp. green cells) by iterating the sampled non-linear dynamics. Moreover, the initial “implicit paving” gradually leads to possibly overlapping cells (see the blue borders in Fig. 6) since the complexity of the polynotope computed at each iteration is reduced to a finite number (50) of generators.
The sparse structure of the exponent matrix E of $\mathbf{x}$ at $k = 5$ is given in Fig. 7. The monomials involve 16 symbolic variables (10 of type unit interval: square marks, 6 of type signed: + marks). There are 90 non-zeros elements. The maximum degree is 3. As expected from the propagation of the initial mixed-encoding of states

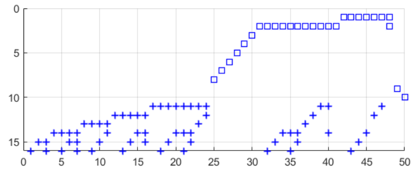

Fig. 7. Sparse structure of the exponent matrix E of $\mathbf{x}$ at $k = 5$: 16 symbolic variables (interval: □ marks, signed: + marks), 50 monomials/generators, 90 non-zero elements.

through a non-linear dynamic, several mixed monomials can be observed, that is, monomials involving both continuous and discrete symbolic variables.

This example of reachability with a mixed-encoding of state sets illustrates the ability of mixed polynotopes to trace the propagation of a significant number of hierarchically organized cells within non-trivial dynamics. Bissections have been avoided by dealing with polynomial dependencies between symbolic variables of different types combining continuous and discrete value domains. Moreover, the ability to explicitly characterize the overlapping between cells forming a partition of the initial state set paves the way for efficient symbolic abstraction techniques.

7.4.2 Non-linear and mixed filtering with PKF

The Polynotopic Kalman Fiter (PKF) developed in section §6 is applied to the modified Lotka-Volterra dynamic without and with a mixed encoding of states:

$$x_+ = x + \mathsf{f}(x, u)h + E\bar{v}, \quad (38)$$
$$y = x_1 + F\bar{w}. \quad (39)$$

The prediction model (38) (resp. measurement equation (39)) correspond to (20) (resp. (21)) in the formulation of PKF with $v = [\bar{v}; \bar{w}]$ and $\Phi = \mathcal{I}$. The step size is $h = 0.04$ and $k \in \{0, \dots, N\} \subset \mathbb{N}$ with $N = 750$ iterations. The initial and final times are $t_0 = 0$ and $t_f = Nh = 30$. The input is $u = 2$ for $t \in [10, 20[$ i.e. $250 \leq k < 500$, and $u = 0$ otherwise. $E = 3.10^{-3}\mathcal{I}$, $F = 1.5$. The state and measurement noises are assumed to be bounded as $\bar{v} \in [-1, +1]^2$ and $\bar{w} \in [-1, +1]$ i.e. $\mathcal{P}\mathbf{v} = [-1, +1]^3$. The initial state set is assumed to be bounded by $\mathcal{P}\mathbf{x}_0 = [[5, 25]; [5, 25]] \subset \mathbb{R}^2$. These bounds are obtained from:

$$\mathbf{x}_{1,0} = 15 + 10.Z_s^g([!(g, \mathtt{s}); !(1, \mathtt{i})]),$$
$$\mathbf{x}_{2,0} = 15 + 10.Z_s^g([!(g, \mathtt{s}); !(1, \mathtt{i})]),$$
$$\mathbf{v} = [Z_s^0(!(1, \mathtt{i})); Z_s^0(!(1, \mathtt{i})); Z_s^g([!(g, \mathtt{s}); !(1, \mathtt{i})])],$$

where g stands for the granularity level of a mixed encoding of the initial states and the measurement noise at each sample time k. Two cases are considered:
i) $g = 0$ corresponds to a purely continuous case (solid lines in Fig. 9) with only symbols of type unit interval.
ii) $g = 2$ corresponds to a mixed case (dashed lines in Fig. 9) involving symbols of different types (signed and interval) in mixed polynotope computations.
At each iteration of PKF, the reduction operator $\downarrow_q$ with $q = 50$ or $q = 100$ is used to: *a)* implement the reduction of $\mathbf{x}$ as in (25), *b)* reduce the product $\mathbf{x}_1\mathbf{x}_2$ in $\mathsf{f}(\mathbf{x}, \mathbf{u})$; this is the only (inclusion preserving) difference between f (20) and $\tilde{f}$ (26) in this example. Notice also that g (21) equals $\tilde{g}$ (27) since the measurement/innovation equation (39) is linear. The simulation of the "true" system is obtained from $x_0 = [22; 8]$ using Heun's method. Consistently with (39), only the first state $x_1 \in \mathbb{R}$ is measured at each sample time k, and the main purpose of PKF is to estimate state bounds $\mathcal{B}\mathbf{x}$ for the state $x \in \mathbb{R}^2$ while minimizing the (predicted) polynotope covariation trace. The table 10 reports the computation times with a Matlab implementation: The mixed encoding does not increase very significantly the computation time in spite of the number of discrete configurations, contrary to the number of generators. The evolution in time of the memory footprint of $\mathbf{x}$ with $\downarrow_{50}$ is also reported in Fig. 8: once again, only a slight increase is observed in the mixed case ($g = 2$) compared to the continuous one ($g = 0$). The simulation results reported in Fig. 9 show a significant improvement of accuracy compared to EZGKF in a purely bounded-error setting which requires around 2.5 s with $q = 200$ generators as in [7]. In particular, PKF shows an enhanced ability to reconstruct x_2 from noisy measurements of x_1. Mixed encoding tends to give results with increased accuracy, especially for x_1, provided the number of generators is sufficient. Meanwhile, the maximum degree of computed polytopes is decreased from 6 (resp. 7) in the continuous case $g = 0$ with $q = 50$ (resp. $q = 100$) to only 4 in both mixed cases i.e. $g = 2$

Table 10
Computation times for 750 iterations of PKF in seconds (s). Cases: continuous ($g = 0$) vs. mixed ($g = 2$), and $\downarrow_{50}$ vs. $\downarrow_{100}$.

	$g = 0$	$g = 2$
$q = 50$	3.4 s	3.8 s
$q = 100$	13.9 s	14.8 s

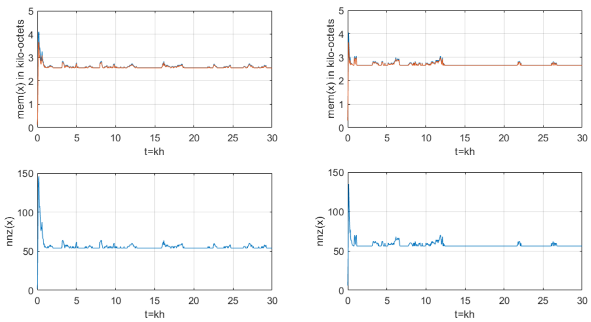

Fig. 8. Memory footprint of the polynotopic state estimate $\mathbf{x}$ with PKF: Evolution in time under $\downarrow_{50}$. Left: Continuous ($g = 0$). Right: Mixed ($g = 2$). Top: Memory size $mem(\mathbf{x})$ in kilo-octets occupied by the object $\mathbf{x}$, depending on whether the sparse exponent matrix E of $\mathbf{x}$ is transposed (red) or not (blue). Bottom: Number of non-zero elements in E.

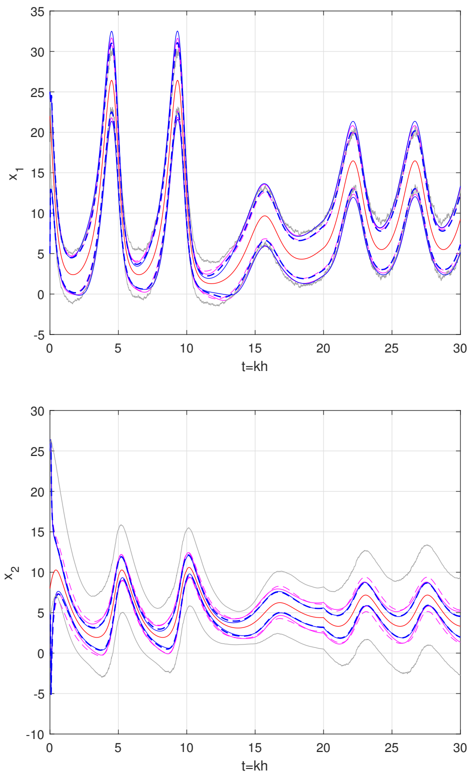

Fig. 9. Estimated state bounds vs. "true" values (red) of x_1 (top) and x_2 (bottom) with the Lotka-Volterra model: PKF with continuous (solid lines) or mixed (dashed lines) polynotopes, and $\downarrow_{50}$ (magenta) or $\downarrow_{100}$ as reduction operator. Comparison with EZGKF in a bounded-error setting (grey).

with $q = 50$ or 100. This is consistent with the reduced size of remainder intervals for $g = 2$ and the influence of rewriting rules, in particular the inclusion neutral rule (12). This illustrates the ability of PKF to efficiently deal with nonlinear and mixed dynamics.

8 Conclusion

An approach for functional sets with typed symbols is introduced in this work. An explicit distinction between syntax and semantics helps formalize the management of dependencies, characterize sources of conservatism and analyze the impact of evaluation strategies (inner first vs. outer first/lazy/call-by-need). Image-sets with typed symbols generalize several set-representations like zonotopes and polynomial zonotopes to mixed domains, as exemplified with mixed polynotopes. The combination of polynomial functions with interval, signed and boolean symbolic variables through simple rewriting rules makes it possible to gather in a single compact and efficient data structure the description of non-convex and non-connected sets which would usually require costly bissection/splitting strategies to be propagated. The mixed-encoding of intervals proposed in this context allows to tune the granularity level of the discrete part of the description and, meanwhile, control the combinatorial complexity through the use of reduction operators. In addition, the traceability of uniquely identified typed symbols paves the way for advanced mixed sensitivity analysis and symbolic abstraction techniques. The reachability results show the relevance of the proposed approach to deal with the verification and synthesis of Cyber-Physical Systems (CPS). Based on modeling tools for nonlinear hybrid systems, a compositional implementation of advanced reachability and filtering algorithms is made possible by simply using operator overloading. This has been exemplified with the proposed Polynotopic Kalman Filter (PKF) which paves the way to advanced hybrid nonlinear filtering techniques preserving inclusion. Moreover, several bridges with random variables and stochastic filtering have been outlined as well as bridges with functional programming and object oriented paradigms, robust (and interpretable?) artificial intelligence [29,12] with the neural network activation function ReLU, sensitivity analysis, and symbolic abstractions of hybrid systems. Much remains to be done to exploit these connections.

Acknowledgements

The author would like to thank Prof. Ali Zolghadri for many insightful discussions during this research work and for his careful reading of the manuscript.

References

[1] Teodoro Alamo, José M. Bravo, and Camacho Eduardo F. Guaranteed state estimation by zonotopes. *Automatica*, 41:1035–1043, 2005.

[2] Matthias Althoff. Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets. In Calin Belta and Franjo Ivancic, editors, *Proceedings of the 16th international conference on Hybrid systems: computation and control, HSCC 2013, April 8-11, 2013, Philadelphia, PA, USA*, pages 173–182. ACM, 2013.

[3] Matthias Althoff and Bruce H. Krogh. Reachability analysis of nonlinear differential-algebraic systems. *IEEE Trans. Autom. Control.*, 59(2):371–383, 2014.

[4] Christophe Combastel. A state bounding observer based on zonotopes. *European Control Conference, ECC 2003*, pages 2589–2594, 2003.

[5] Christophe Combastel. A state bounding observer for uncertain non-linear continuous-time systems based on zonotopes. *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference, CDC-ECC '05*, 2005:7228–7234, 2005.

[6] Christophe Combastel. Zonotopes and Kalman observers: Gain optimality under distinct uncertainty paradigms and robust convergence. *Automatica*, 55:265–273, May 2015.

[7] Christophe Combastel. An extended zonotopic and gaussian kalman filter (EZGKF) merging set-membership and stochastic paradigms: Toward non-linear filtering and fault detection. *Annual Reviews in Control*, 42:232–243, 2016.

[8] Christophe Combastel and Sid-Ahmed Raka. On computing envelopes for discrete-time linear systems with affine parametric uncertainties and bounded inputs. *IFAC Proc. Volumes (IFAC-PapersOnline)*, 44:4525–4533, 2011.

[9] Christophe Combastel and Ali Zolghadri. A distributed Kalman filter with symbolic zonotopes and unique symbols provider for robust state estimation in CPS. *International Journal of Control*, (0):1–17, 2019.

[10] Thach Ngoc Dinh, Frédéric Mazenc, Zhenhua Wang, and Tarek Raïssi. On fixed-time interval estimation of discrete-time nonlinear time-varying systems with disturbances. In *2020 American Control Conference, ACC 2020, Denver, CO, USA, July 1-3, 2020*, pages 2605–2610. IEEE, 2020.

[11] Luiz Henrique De Figueiredo and Jorge Stolfi. Affine arithmetic: Concepts and applications. *Numerical Algorithms*, 37(1-4):147–158, 2004.

[12] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. AI2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, May 2018.

[13] Antoine Girard. Reachability of uncertain linear systems using zonotope. In *Hybrid Systems: Comput. and Control*, volume 3414 of *LNCS*, pages 291–305. Springer, 2005.

[14] Eric Goubault and Sylvie Putot. A zonotopic framework for functional abstractions. *Formal Methods in System Design*, 47(3):302–360, 2015.

[15] Eric Goubault and Sylvie Putot. Robust under-approximations and application to reachability of non-linear control systems with disturbances. *IEEE Control. Syst. Lett.*, 4(4):928–933, 2020.

[16] Branko Grünbaum. *Convex Polytopes*. Springer New York, New York, 2003.

[17] Fabian Immler, Matthias Althoff, Xin Chen, Chuchu Fan, Goran Frehse, Niklas Kochdumper, Yangge Li, Sayan Mitra, Mahendra Singh Tomar, and Majid Zamani. ARCH-COMP18 cat. report: Cont. and hybrid sys. with nonlinear dyn. In G. Frehse, M. Althoff, S. Bogomolov, and T. Johnson, editors, *ARCH18. 5th Int. Workshop on Applied Verification of Continuous and Hybrid Systems, ARCH@ADHS 2018, Oxford, UK, July 13, 2018*, volume 54 of *EPiC Series in Computing*, pages 53–70. EasyChair, 2018.

[18] Luc Jaulin, Michel Kieffer, Olivier Didrit, and Eric Walter. *Applied interval analysis*. Springer, 2001.

[19] Rudolf E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering*, 82:35–45, 1960.

[20] Niklas Kochdumper and Matthias Althoff. Representation of polytopes as polynomial zonotopes. *arXiv:1910.07271*.

[21] Niklas Kochdumper and Matthias Althoff. Constrained polynomial zonotopes. *arXiv:2005.08849*, 2020.

[22] Niklas Kochdumper and Matthias Althoff. Sparse polynomial zonotopes: A novel set representation for reachability analysis. *IEEE Transactions on Automatic Control*, pages 1–1, 2020.

[23] Wolfgang Kühn. Rigorously computed orbits of dynamical systems without the wrapping effect. *Computing*, 61:pp. 47–67, 1998.

[24] Alex Kurzhanskiy and Pravin Varaiya. Ellipsoidal techniques for reachability analysis of discrete-time linear systems. *IEEE Trans. on Automatic Control*, 52(1):26–38, 2007.

[25] Vu Tuan Hieu Le, Cristina Stoica, Teodoro Alamo, Eduardo F. Camacho, and Didier Dumur. Zonotopic guaranteed state estimation for uncertain systems. *Automatica*, 49:3418–3424, 2013.

[26] Moussa Maïga, Nacim Ramdani, Louise Travé-Massuyès, and Christophe Combastel. A comprehensive method for reachability analysis of uncertain nonlinear hybrid systems. *IEEE Trans. Autom. Control.*, 61(9):2341–2356, 2016.

[27] Kyoko Makino and Martin Berz. Rigorous integration of flows and odes using taylor models. In Hiroshi Kai, Hiroshi Sekigawa, Tateaki Sasaki, Kiyoshi Shirayanagi, and Ilias S. Kotsireas, editors, *Symbolic Num. Comput., SNC '09, Kyoto, Japan - August 03 - 05, 2009*, pages 79–84. ACM, 2009.

[28] Pierre-Jean Meyer, Alex Devonport, and Murat Arcak. TIRA : Toolbox for interval reachability analysis. In *Proc. of the 22nd ACM Int. Conf. on Hybrid Systems: Computation and Control*. ACM, apr 2019.

[29] Matthew Mirman, Timon Gehr, and Martin T. Vechev. Differentiable abstract interpretation for provably robust neural networks. In Jennifer G. Dy and Andreas Krause, editors, *Proc. of the 35th Int. Conf. on Machine Learning, ICML 2018, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3575–3583. PMLR, 2018.

[30] Ian M. Mitchell. The flexible, extensible and efficient toolbox of level set methods. *J. Sci. Comput.*, 35(2-3):300–329, 2008.

[31] Luis Orihuela, Pablo Millán, Samira Roshany-Yamchi, and Ramón A. García. Negotiated distributed estimation with guaranteed performance for bandwidth-limited situations. *Automatica*, 87:94–102, 2018.

[32] Benjamin C. Pierce. *Types and Programming Languages*. The MIT Press, 2002.

[33] Masoud Pourasghar, Christophe Combastel, Vicenç Puig, and Carlos Ocampo-Martinez. FD-ZKF: A zonotopic Kalman filter optimizing fault detection rather than state estimation. *Journal of Process Control*, 73:89–102, 2019.

[34] Vicenç Puig, Jordi Saludes, and Joseba Quevedo. Worst-case simulation of discrete linear time-invariant interval dynamic systems. *Reliable Computing*, 9(4):251–290, 2003.

[35] Tarek Raïssi and Denis V. Efimov. Some recent results on the design and implementation of interval observers for uncertain systems. *Automatica*, 66(3):213–224, 2018.

[36] Nacim Ramdani, Nacim Meslem, and Yves Candau. A hybrid bounding method for computing an over-approximation for the reachable set of uncertain nonlinear systems. *IEEE Trans. Autom. Control.*, 54(10):2352–2364, 2009.

[37] Siegfried M. Rump and Masahide Kashiwagi. Implementation and improvements of affine arithmetic. *Nonlinear Theory and Its Applications, IEICE*, 6(3):341–359, 2015.

[38] Joseph K. Scott, Davide M. Raimondo, Giuseppe Roberto Marseglia, and Richard D. Braatz. Constrained zonotopes: A new tool for set-based estimation and fault detection. *Automatica*, 69:126–136, jul 2016.

[39] François-Régis Sinot. Complete laziness: A natural semantics. *Electronic Notes in Theoretical Computer Science*, 204:129–145, apr 2008.

[40] Ye Wang, Sorin Olaru, Giorgio Valmorbida, Vicenç Puig, and Gabriela Cembraño. Set-invariance characterizations of discrete-time descriptor systems with application to active mode detection. *Automatica*, 107:255–263, 2019.

[41] Feng Xu, Vicenç Puig, Carlos Ocampo-Martinez, Sorin Olaru, and Florin Stoican. Set-theoretic methods in robust detection and isolation of sensor faults. *Int. J. Syst. Sci.*, 46(13):2317–2334, 2015.