Polynomial Interrupt Timed Automata

Béatrice Bérard[1], Serge Haddad[2], Claudine Picaronny[2], Mohab Safey El Din[1], and Mathieu Sassolas[3]

[1] Sorbonne Université, Université P. & M. Curie, LIP6, CNRS UMR 7606, Paris, France
[2] École Normale Supérieure de Cachan, LSV, CNRS UMR 8643, INRIA, Cachan, France
[3] Université Paris-Est, LACL, Créteil, France

Abstract. Interrupt Timed Automata (ITA) form a subclass of stopwatch automata where reachability and some variants of timed model checking are decidable even in presence of parameters. They are well suited to model and analyze real-time operating systems. Here we extend ITA with polynomial guards and updates, leading to the class of polynomial ITA (PolITA). We prove the decidability of the reachability and model checking of a timed version of CTL by an adaptation of the *cylindrical decomposition* method for the first-order theory of reals. Compared to previous approaches, our procedure handles parameters and clocks in a unified way. Moreover, we show that PolITA are incomparable with stopwatch automata. Finally additional features are introduced while preserving decidability.

1 Introduction

Hybrid Automata. Hybrid systems [16] combine continuous evolution of variables according to flow functions (described by differential inclusions) in control nodes, and discrete jumps between these nodes, where the variables can be tested by guards and updated. This class of models is very expressive and all relevant verification questions (*e.g.* reachability) are undecidable. For the last twenty years, a large amount of research was devoted to identifying subclasses with decidable properties, by restricting the continuous dynamics and/or the discrete behavior of the systems. Among these classes lie the well known Timed Automata (TA) [3], where all variables are *clocks* (with derivative $\dot{x} = 1$), guards are comparisons of clocks with rational constants, and updates are resets. It is proved in [17] that reachability becomes undecidable when adding one stopwatch (with $\dot{x} = 1$ or $\dot{x} = 0$) to timed automata. Decidability results were also obtained for larger classes (see [5,2,17,20,4]), usually by building from the associated transition system (with uncountable state space) a finite abstraction preserving a specific class of properties, like reachability or those expressed by temporal logic formulas. In all these abstractions, a state is a pair composed of a control node and a polyhedron of variable values. Examples of such classes include initialized rectangular automata [17] where $\dot{x} \in [a, b]$ or o-minimal hybrid systems [20] where the flow is more general, for instance of the form $\dot{x} = Ax$ over $\mathbb{R}^n$ for some matrix A. In both cases, the variables must be (possibly non deterministically) reinitialized at discrete jumps.

Interrupt Timed Automata. The class of Interrupt Timed Automata (ITA), incomparable with TA, was introduced in [8,10] as another subclass of hybrid automata with a (time-abstract) bisimulation providing a finite quotient, thus leading to decidability of reachability and some variants of timed model checking. In a basic n-dimensional ITA, control nodes are organized along n levels, with n stopwatches (also called clocks hereafter), one per level. At a given level, the associated clock is active, while clocks from lower levels are frozen and clocks from higher levels are irrelevant. Guards are linear constraints and the clocks can be updated by linear expressions (using only clocks from lower levels). The particular hierarchical structure of ITA makes them particularly well suited for modeling systems with interruptions, like real-time operating systems. ITA were extended with parameters in [9] while preserving decidability by combining the finite abstraction of original ITA with a finite partition of parameter values.

Contribution. We define the class PolITA, of polynomial ITA, where linear expressions on clocks are replaced by polynomials with rational coefficients both for guards and updates. For instance, a guard at level 2 with clock x_2 can be of the form $P_1(x_1)x_2^2 + P_2(x_1) \geq 0$, where P_1 and P_2 are polynomials with single variable x_1, the clock of level 1. Thus, guards are more expressive than in the whole class of linear hybrid automata. Such guards can be useful for instance if some objects are produced at given levels, and operations on higher levels on these objects require polynomial-time computations w.r.t. the size of these objects. In addition, such guards can simulate irrational (algebraic) constraints, a case that becomes undecidable in the setting of timed automata [21].

We establish that model checking of a timed extension of CTL (which contains reachability) is decidable in 2EXPTIME for PolITA by adapting the cylindrical decomposition [13,6] related to the first order theory of reals. This decomposition produces a finite partition of the state space, which is the basis for the construction of a finite bisimulation quotient. The first order theory of reals has already been used in several works on hybrid automata [20,4] but it was restricted to the dynamical part, with discrete jumps that must reinitialize the variables (like in o-minimal hybrid systems). Our adaptation consists in an on-the-fly construction avoiding in the favorable cases to build the whole decomposition.

From an expressiveness point of view, we show that (contrary to ITA) PolITA are incomparable with stopwatch automata (SWA). Finally, we prove that the decidability result still holds with several extensions: adding auxiliary clocks and parameters, and enriching the possible updates. In particular, parametric ITA [9] can be seen as a subclass of PolITA, and the complexity of our reachability algorithm is better than [9] (2EXPSPACE).

Outline. We describe the model of polynomial ITA in Section 2, with an example and the presentation of the model checking problem. In Section 3, we revisit and adapt in this context the cylindrical decomposition for the first theory of reals, with a special focus on the related algorithmic questions. The decision procedure for the model checking problem in PolITA is then presented in Section 4, with an example of the construction. Finally, we describe several extensions in Section 5 and conclude in Section 6.

2 Polynomial ITA

2.1 Definition

Let $\mathbb{N}$ denote the set of natural numbers, $\mathbb{Z}$ the set of integers, $\mathbb{Q}$ the set of rationals, and $\mathbb{R}$ the set of real numbers, with $\mathbb{R}_{\geq 0}$ the set of non negative real numbers.

Let $X = \{x_1, \ldots, x_n\}$ be a finite set of n variables called clocks. We write $\mathbb{Q}[x_1, \ldots, x_n]$ for the set of polynomials with n variables and rational coefficients. A *polynomial constraint* is a conjunction of constraints of the form $P \bowtie 0$ where $P \in \mathbb{Q}[x_1, \ldots, x_n]$ and $\bowtie \in \{<, \leq, =, \geq, >\}$, and we denote by $\mathcal{C}(X)$ the set of polynomial constraints. We also define $\mathcal{U}(X)$, the set of *polynomial updates* over X as:

$$\mathcal{U}(X) = \left\{ \bigwedge_{x \in X} x := P_x \;\middle|\; \forall x,\, P_x \in \mathbb{Q}[x_1, \ldots, x_n] \right\}.$$

A valuation for X is a mapping $v \in \mathbb{R}^X$, sometimes also identified to the vector $(v(x_1), \ldots, v(x_n)) \in \mathbb{R}^n$. The valuation where $v(x) = 0$ for all $x \in X$ is denoted by $\mathbf{0}$. For $P \in \mathbb{Q}[x_1, \ldots, x_n]$ and v a valuation, the value of P at v is $P(v) = P(v(x_1), \ldots, v(x_n))$. A valuation v satisfies the constraint $P \bowtie 0$, written $v \models P \bowtie 0$, if $P(v) \bowtie 0$. The notation is naturally extended to a polynomial constraint: $v \models \varphi$ with $\varphi = \bigwedge_i P_i \bowtie_i 0$ if $v \models P_i \bowtie_i 0$ for every i.

An update of valuation v by $u = \wedge_{x \in X} x := P_x \in \mathcal{U}(X)$ is the valuation $v[u]$ defined by $v[u](x) = P_x(v)$ for every $x \in X$. Hence an update is atomic in the sense that all variables are set at the same time: the new value of variables depend on the old values of v.

For a valuation v and a delay $d \in \mathbb{R}_{\geq 0}$, the valuation $v' = v +_k d$, corresponding to *time elapsing for clock* x_k, is defined by $v'(x_k) = v(x_k) + d$ and $v'(x) = v(x)$ for any other clock x.

Definition 2.1 (PolITA). *A* polynomial interrupt timed automaton (PolITA) *is a tuple* $\mathcal{A} = \langle \Sigma, Q, q_0, F, X, \lambda, \Delta \rangle$*, where:*

- Σ *is a finite alphabet,*
- Q *is a finite set of states,* q_0 *is the initial state,* $F \subseteq Q$ *is the set of final states,*
- $X = \{x_1, \ldots, x_n\}$ *consists of* n *interrupt clocks,*

– *the mapping $\lambda : Q \to \{1, \ldots, n\}$ associates with each state its level and $x_{\lambda(q)}$ is called the* active clock *in state q.*
– *$\Delta \subseteq Q \times \mathcal{C}(X) \times (\Sigma \cup \{\varepsilon\}) \times \mathcal{U}(X) \times Q$ is the set of transitions. Let $q \xrightarrow{\varphi,a,u} q'$ in Δ be a transition with $k = \lambda(q)$ and $k' = \lambda(q')$. The guard φ is a conjunction of constraints $P \bowtie 0$ with $P \in \mathbb{Q}[x_1, \ldots, x_k]$ (P is a polynomial over clocks from levels less than or equal to k). The update u is of the form $\wedge_{i=1}^{n} x_i := C_i$ with:*
  - *if $k > k'$, i.e. the transition decreases the level, then for $1 \leq i \leq k'$, $C_i = x_i$ and for $i > k'$, $C_i = 0$;*
  - *if $k \leq k'$ then for $1 \leq i < k$, $C_i = x_i$, $C_k = P$ for $P \in \mathbb{Q}[x_1, \ldots, x_{k-1}]$ or $C_k = x_k$, and for $i > k$, $C_i = 0$.*

Remark that although it is possible to compare an active clock in a non-polynomial way, *e.g.* $x_2 > \sqrt{x_1}$ (which can be translated as $x_2^2 > x_1 \wedge x_1 \geq 0$), it cannot be updated in such a fashion.

Example 2.2. PoLITA $\mathcal{A}_0$ of Fig. 1 has two levels, with q_0 at level 1 and q_1 and q_2 at level 2, with q_2 the single final state. At level 1, only x_1 appears in guards and updates (here the only update is the resetting of x_1 by action a'), while at level 2 guards use polynomials in both x_1 and x_2.

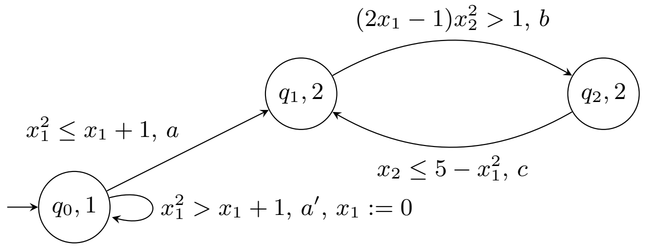


Fig. 1. A sample PoLITA $\mathcal{A}_0$.

A *configuration* (q, v) consists of a state q of $\mathcal{A}$ and a clock valuation v.

Definition 2.3. *The semantics of a* PoLITA *$\mathcal{A}$ is defined by the (timed) transition system $\mathcal{T}_\mathcal{A} = (S, s_0, \to)$, where $S = \{(q, v) \mid q \in Q,\ v \in \mathbb{R}^X\}$ is the set of configurations, with initial configuration $s_0 = (q_0, \mathbf{0})$. The relation $\to$ on S consists of two types of steps:*

Time steps: *Only the active clock in a state can evolve, all other clocks are frozen. For a state q with active clock $x_{\lambda(q)}$, a time step of duration $d \in \mathbb{R}_{\geq 0}$ is defined by $(q, v) \xrightarrow{d} (q, v')$ with $v' = v +_{\lambda(q)} d$. A time step of duration 0 leaves the system $\mathcal{T}_\mathcal{A}$ in the same configuration.*

Discrete steps: *There is a discrete step $(q, v) \xrightarrow{a} (q', v')$ whenever there exists a transition $q \xrightarrow{\varphi,a,u} q'$ in Δ such that $v \models \varphi$ and $v' = v[u]$.*

An run of a PolITA $\mathcal{A}$ is a path in $\mathcal{T}_{\mathcal{A}}$. The *trace* of a run is the sequence of letters (or word) appearing in the path. The *timed word* is the sequence of letters along with the absolute time of the occurrence, *i.e.* the sum of all delays appearing before the letter. Given a subset $F \subseteq Q$ of final states, a run is accepting if it ends in a state of F. This defines the *language* (resp. *timed language*) as the set of traces (resp. timed words) of accepting runs.

Example 2.4. The PolITA $\mathcal{A}_0$ can only take the transition from q_0 to q_1 before x_1 reaches $\frac{1+\sqrt{5}}{2}$, *i.e.* at the point where the red curve crosses the x_1 axis on Fig. 2. Then, transition b from q_1 to q_2 can only be taken once x_2 reaches the grey areas. Transition c cannot however be taken once the green curve has been crossed. Hence the loop bc can be taken as long as the clocks remain in the dark gray zone. In the sequel, we show how to symbolically compute these zones. Since q_2 is a final state, the run depicted in Fig. 2 is accepted by $\mathcal{A}$. The associated timed word is $(a, 1.2)(b, 2.3)(c, 2.6)(b, 3.3)(c, 3.9)(b, 5.1)$, and the trace is the word $abcbcb$.

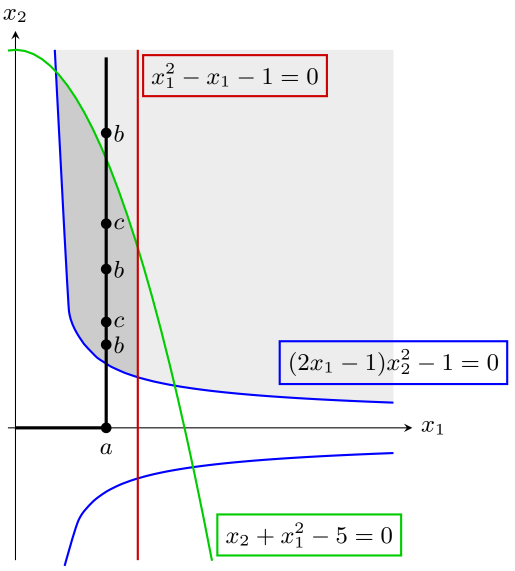


Fig. 2. A trajectory of clocks of $\mathcal{A}_0$ in the 2-dimensional plane.

2.2 Verification problems for PolITA

Given a PolITA $\mathcal{A}$, natural questions arise regarding its behavior. The most standard one is the *reachability problem* which is the decision problem asking whether a given state can be reached from the initial configuration. This allows in particular to decide whether the timed language is nonempty, which is equivalent to testing the reachability of a final state.

More elaborate queries regarding the behavior of a PoLITA can be expressed through temporal logics like CTL [15,22] or timed extensions of such logics like TCTL [1,18]. Here we use a timed extension of CTL which allows to reason over the values of clocks of the PoLITA.

Let AP be a set of atomic propositions, we equip the states of $\mathcal{A}$ with a labeling $lab : Q \to 2^{AP}$ of propositions that hold in the given state. For convenience, we assume that $Q \subseteq AP$ with for all $q, q' \in Q$, $q' \in lab(q)$ iff $q = q'$.

Definition 2.5. *Formulas of the timed logic* TCTL_{int} *are defined by the following grammar:*

$$\psi ::= p \mid \psi \wedge \psi \mid \neg\psi \mid P \bowtie 0 \mid \mathsf{A}\,\psi\,\mathsf{U}\,\psi \mid \mathsf{E}\,\psi\,\mathsf{U}\,\psi$$

where $p \in AP$, P *is a polynomial of* $\mathbb{Q}[x_1, \dots, x_n]$, *and* $\bowtie \in \{>, \geq, =, \leq, <\}$.

We use the classical shorthands $\mathsf{F}p = \mathit{true}\,\mathsf{U}\,p$, $\mathsf{G}p = \neg\mathsf{F}\neg p$, and boolean operators. The reachability problem of a state q is simply the satisfaction of $\mathsf{E}\,\mathsf{F}q$.

The formulas of $\mathsf{TCTL}_{\mathrm{int}}$ are interpreted over configurations of $\mathcal{A}$, hence the semantics of $\mathsf{TCTL}_{\mathrm{int}}$ is defined as follows on the transition system $\mathcal{T}_{\mathcal{A}}$ associated with $\mathcal{A}$. Let $Run(s)$ denote all runs starting from configuration $s = (q, v)$. For $\rho = (q, v) \xrightarrow{d_1} (q, v +_{\lambda(q)} d_1) \xrightarrow{a_1} (q_2, v_2) \cdots \in Run(s)$, a position in ρ is a pair $\pi = (i, \delta)$ where $1 \leq i$ and $0 \leq \delta \leq d_i$. The configuration corresponding to π is $s_\pi = (q_i, v_i +_{\lambda(q_i)} \delta)$ (with $q_1 = q$ and $v_1 = v$). We denote by $<_\rho$ the strict lexicographical order over positions of ρ.
For basic formulas:

$$\begin{array}{ll} s \models p & \text{iff } p \in lab(s) \\ s \models P \bowtie 0 & \text{iff } v \models P \bowtie 0 \end{array}$$

and inductively:

$$\begin{array}{ll} s \models \varphi \wedge \psi & \text{iff } s \models \varphi \text{ and } s \models \psi \\ s \models \neg\varphi & \text{iff } s \not\models \varphi \\ s \models \mathsf{A}\,\varphi\,\mathsf{U}\,\psi & \text{iff for all } \rho \in Run(s),\ \rho \models \varphi\,\mathsf{U}\,\psi \\ s \models \mathsf{E}\,\varphi\,\mathsf{U}\,\psi & \text{iff there exists } \rho \in Run(s) \text{ s. t. } \rho \models \varphi\,\mathsf{U}\,\psi \\ \text{with } \rho \models \varphi\,\mathsf{U}\,\psi & \text{iff there is a position } \pi \in \rho \text{ s. t. } s_\pi \models \psi \\ & \quad \text{and } \forall \pi' <_\rho \pi,\ s_{\pi'} \models \varphi \vee \psi. \end{array}$$

The automaton $\mathcal{A}$ satisfies ψ (written $\mathcal{A} \models \psi$) if the initial configuration s_0 of $\mathcal{T}_{\mathcal{A}}$ satisfies ψ. The *model checking* problem asks, given $\mathcal{A}$ and ψ, whether $\mathcal{A} \models \psi$.

As mentioned in the introduction, an exhaustive traversal of the (uncountable) transition system $\mathcal{T}_{\mathcal{A}}$ is not possible, and the model checking algorithm relies on an abstraction of said transition system. This abstraction needs to be refined enough to capture both time elapsing and discrete jumps through the crossing of a transition. Namely, two configurations in the same abstraction class should reach the same successor classes when time elapses or when an update is applied. Moreover, the truth value of subformulas $P \bowtie 0$ should be invariant in each abstraction class.

The previous works of [8,10,9] on ITA built such an abstraction by relying on a set of *expressions* with rational coefficients. These expressions contained linear forms involved in guards and updates, along with the active clock of the level. Moreover, since the ordering of two expressions at a given level could rely on the value of lower-level clocks, some expressions were required at inferior levels. The classes were then defined as subsets of $\mathbb{R}^n$ where the ordering of expressions was constant.

In the sequel, we adapt the above process in the context of PolITA, where the constraints are polynomial rather than linear, and hence yield regions that are not polyhedra, but cells defined by a so called *cylindrical decomposition.*

3 Cylindrical algebraic decomposition for first-order theory of reals

Cylindrical algebraic decomposition is introduced by Collins in [13] for solving quantifier elimination problems of first-order formulas over the reals. The first algorithm for solving this problem was given by Tarski in [23] but its complexity was non elementary recursive. Cylindrical algebraic decomposition is doubly exponential in the number of variables and is now a popular technique for solving polynomial systems over the reals. Given a polynomial family, it essentially partionates the ambient space into *cells* which are homeomorphic to $]0,1[^i$ over which the input is sign-invariant. These cells are also intrinsically arranged together with a nice cylindrical structure which we explain further. Later on, a procedure in EXPSPACE was established [7]. The best lower bound currently known for this problem is $STA(*, 2^{n^{O(1)}}, n)$ (a complexity class defined by machines with limited alternations and located between EXPTIME and EXPSPACE) and it already holds without the multiplication [11].

We consider formulas that express properties of reals. There are inductively defined as follows. An arithmetic expression is:

- either an integer constant, a variable;
- or $e_1 + e_2$, $e_1 * e_2$ where e_1 and e_2 are arithmetic expressions.

A formula is:

- a basic formula: $e \sim 0$ where $\sim \in \{<, =\}$ and e is an arithmetic expression;
- or $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\neg\varphi_1$, $\forall x \varphi_1$, $\exists x \varphi_1$ where φ_1 and φ_2 are formulas and x is a variable.

A sentence is a formula without free variables. A sentence has a truth value when interpreted over $\mathbb{R}$ and we are looking for deciding the truth of a formula.

For our purposes, we will adapt the cylindrical algebraic decomposition. So we develop in the section all the required machinery. Here we only describe the general principles and we explain how it can be used for deciding the truth of a formula. The first concept that we introduce is the one of *cell.*

Definition 3.1. *A cell of level n is a subset of $\mathbb{R}^n$ inductively defined as follows.*

– *When $n = 1$, it is either a point or an open interval.*
– *A cell C of level $n + 1$ is based on a cell C' of level n. It has one of the following shapes.*
  1. *$C = \{(x, f(x)) \mid x \in C'\}$ with f a continuous function from C' to $\mathbb{R}$;*
  2. *$C = \{(x, y) \mid x \in C' \wedge l(x) < y < u(x)\}$ with $l < u$ continuous functions from C' to $\mathbb{R}$, possibly with $l = -\infty$ and/or $u = +\infty$.*

By convention the single cell of level 0 is $\mathbb{R}^0$.

Let $\mathcal{P} = \{\mathcal{P}_i\}_{1 \leq i \leq n}$ be a family of subsets of polynomials such that for all $P \in \mathcal{P}_i$, $P \in \mathbb{R}[X_1, \ldots, X_i]$. By convention, we extend $\mathcal{P}$ with $\mathcal{P}_0 = \emptyset$. The second concept that we introduce is the *sign invariance* of a cell w.r.t. $\mathcal{P}$.

Definition 3.2. *Let $\mathcal{P} = \{\mathcal{P}_i\}_{i \leq n}$. A cell C of level i is $\mathcal{P}$-invariant if:*

– *For all $j \leq i$, for all $P \in \mathcal{P}_j$, for all $x, y \in C$ $sign(P(x)) = sign(P(y))$.*
– *When $i < n$,*
  1. *either $C \times \mathbb{R}$ is $\mathcal{P}$-invariant;*
  2. *or there exists $f_1 < \ldots < f_r$ continuous functions from C to $\mathbb{R}$ such that all the following cells are $\mathcal{P}$-invariant:*

     • *for all $1 \leq i \leq r$, $\{(x, f_i(x)) \mid x \in C\}$;*
     • *for all $0 \leq i \leq r$, $\{(x, y) \mid x \in C \wedge f_i(x) < y < f_{i+1}(x)\}$ with the convention that $f_0 = -\infty$ and $f_{r+1} = +\infty$.*

Observe that $\mathbb{R}^0$ is $\mathcal{P}$-invariant, and that one can inductively define a tree of $\mathcal{P}$-invariant cells as follows.

– The root of the tree is $\mathbb{R}^0$;
– Let C be a $\mathcal{P}$-invariant cell of level $i < n$ belonging to the tree. Then depending on the kind of invariance,
  1. either C has a single child $C \times \mathbb{R}$;
  2. or for some $r \in \mathbb{N}\setminus\{0\}$, C has $2r+1$ ordered children $\{(x, y) \mid x \in C \wedge y < f_1(x)\}$, $\{(x, f_1(x)) \mid x \in C\}$, $\{(x, y) \mid x \in C \wedge f_1(x) < y < f_2(x)\}$, $\ldots$, $\{(x, y) \mid x \in C \wedge y > f_r(x)\}$.

This tree is also called a *cylindrical decomposition.*

Example 3.3 ([6]). Consider the single polynomial $P_s = X_1^2 + X_2^2 + X_3^2 - 1$, with $P_s = 0$ representing a sphere of radius 1 in $\mathbb{R}^3$, as shown in Fig. 3. At level 1, $\mathbb{R}$ is partitioned into 5 cells:

$$C_{-\infty} =]-\infty, -1[\qquad C_{-1} = \{-1\} \qquad C_0 =]-1, 1[$$

$$C_1 = \{1\} \qquad C_{+\infty} =]1, +\infty[$$

At level 2, $\mathbb{R}^2$ is partitioned above the previous cells. There is a single cell $C_{-\infty} \times \mathbb{R}$ above $C_{-\infty}$ (and similarly $C_{+\infty} \times \mathbb{R}$ above $C_{+\infty}$). Above C_{-1} are three cells, its children in the tree:

$$\{-1\} \times]-\infty, 0[\qquad \{(-1, 0)\} \qquad \{-1\} \times]0, +\infty[$$

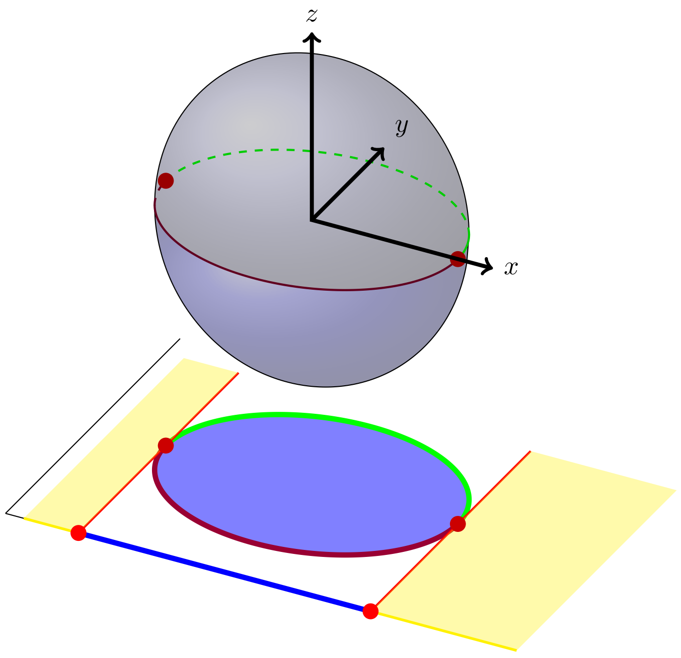


Fig. 3. Cylindrical decomposition of a sphere.

The cells above C_1 are similar.

And above C_0 are 5 cells: the interior of the disc $C_{0,0}$, its lower and upper edges $C_{0,-1}$ and $C_{0,1}$ and the exterior of the circle (the lower and upper parts) $C_{0,-\infty}$ and $C_{0,+\infty}$:

$$C_{0,1} : \begin{cases} -1 < x_1 < 1 \\ x_2 = \sqrt{1 - x_1^2} \end{cases} \qquad C_{0,-1} : \begin{cases} -1 < x_1 < 1 \\ x_2 = -\sqrt{1 - x_1^2} \end{cases}$$

$$C_{0,+\infty} : \begin{cases} -1 < x_1 < 1 \\ x_2 > \sqrt{1 - x_1^2} \end{cases} \qquad C_{0,-\infty} : \begin{cases} -1 < x_1 < 1 \\ x_2 < -\sqrt{1 - x_1^2} \end{cases}$$

$$C_{0,0} : \begin{cases} -1 < x_1 < 1 \\ -\sqrt{1 - x_1^2} < x_2 < \sqrt{1 - x_1^2} \end{cases}$$

At level 3, cell $C_{0,-1}$ is further lifted in three cells where $C_{0,-1,0}$ is half the equator circle of the sphere:

$$C_{0,-1,-\infty} : \begin{cases} -1 < x_1 < 1 \\ x_2 = -\sqrt{1 - x_1^2} \\ x_3 < 0 \end{cases} \qquad C_{0,-1,0} : \begin{cases} -1 < x_1 < 1 \\ x_2 = -\sqrt{1 - x_1^2} \\ x_3 = 0 \end{cases}$$

$$C_{0,-1,+\infty} : \begin{cases} -1 < x_1 < 1 \\ x_2 = -\sqrt{1 - x_1^2} \\ x_3 > 0 \end{cases}$$

And $C_{0,0}$ is lifted into 5 cells: below (and above) the inferior (resp. superior) half of the sphere, said inferior (resp. superior) half, and the interior of the sphere. These cells are determined by two functions $f_1(x_1, x_2) = -\sqrt{1 - x_1^2 - x_2^2}$ and $f_2(x_1, x_2) = \sqrt{1 - x_1^2 - x_2^2}$.

Algorithm 1: Checking the truth of a formula

```
Data: A cylindrical decomposition having parameter $C$ as an element.
Check($\varphi, i, C, \mathcal{S}$): a boolean
Input: $\varphi$, a prenex sentence with $n$ variables, $C$, a $\mathcal{P}$-invariant cell of level $i$
Input: $\mathcal{S}$, a set of pairs of polynomials and signs
Output: the truth value of $\varphi$
Data: $j, k$, some indices
// The expression $sign(P(C))$ uses the sign invariance of $C$
$\mathcal{S} \leftarrow \mathcal{S} \cup \{(P, sign(P(C)) \mid P \in \mathcal{P}_i\}$
// When $i = n$, all atomic formulas of $\psi$ are determined by $\mathcal{S}$
if $i = n$ then return $\psi(\mathcal{S})$
// Let $C_1, \ldots, C_k$ be the children of $C$
if $Q_{i+1} = \exists$ then
    for $j$ from 1 to $k$ do
        if Check($\varphi, i+1, C_j, \mathcal{S}$) then return true
    end
    return false
else
    for $j$ from 1 to $k$ do
        if $\neg$Check($\varphi, i+1, C_j, \mathcal{S}$) then return false
    end
    return true
end
```

Let us explain how a cylindrical decomposition is useful for first-order theory of reals. Any sentence can be transformed into an equivalent prenex formula $\varphi = Q_1 x_1 \ldots Q_n x_n \psi$ such that $Q_i \in \{\forall, \exists\}$ and ψ is a quantifier free formula that checks signs of polynomials evaluated on some of the x_i's. Thus by syntactical examination, we first build the family $\mathcal{P}$ from the polynomials occurring in ψ. Assume that we produce a cylindrical decomposition for $\mathcal{P}$. Then Algorithm 1 solves the decision problem with the call $\texttt{Check}(\varphi, 0, \mathbb{R}^0, \emptyset)$. The correctness of the algorithm is proved by (1) the sign invariance of the cells, (2) the partition of $C \times \mathbb{R}$ between the children of a cell C and (3) a backward inductive property: given a cell C of level i, the truth of $Q_{i+1}x_{i+1} \dots Q_n x_n \psi$ does not depend on the point $(x_1, \dots, x_i) \in C$.

The section is organized as follows. In subsection 3.1 we develop algorithms for rings with some additional assumptions that depend on the algorithms (also presented in [6]). The main hypothesis is that we consider subrings of $\mathbb{R}$ for which there is a decision procedure for evaluation of the sign of an item. In Subsection 3.2, we introduce triangular systems which are representations of algebraic reals and domains of $\mathbb{R}$ and we establish that they are sign-effective. Subsection 3.3 is devoted to the building of a cylindrical decomposition. It consists in two stages: the elimination stage that enlarges $\mathcal{P}$ and the lifting stage that builds the cylindrical decomposition. In this decomposition a cell is represented by an algebraic real (i.e. a triangular system) belonging to it.

3.1 Algorithms in sign-effective subrings of reals

Preliminary remarks. Let us denote by $\mathbb{A}$ a domain *i.e.*, a ring with no divisors of zero. $\mathbb{F}_\mathbb{A}$ denotes the field of fractions of $\mathbb{A}$. Whenever we will describe algorithms involving a domain $\mathbb{A}$, we assume a representation of an item of $\mathbb{A}$. For instance, the representation of $\frac{p}{q} \in \mathbb{Q}$ could be the pair of integers (p, q). We do not require that the representation is unique but that the following operations are effective: addition, multiplication and zero-test. We denote multiplication and addition as usual. The function that performs the zero-test is denoted $\texttt{Null}(\mathbb{A}, d)$ with d, a representation of some item of $\mathbb{A}$.

The goal of this section is to exhibit some problems that can be solved in $\mathbb{A}[X]$ (for $\mathbb{A} \subseteq \mathbb{R}$) when, in addition to the previous operations, the sign of an element of $\mathbb{A}$ can be determined. The sign is defined by $\mathit{sign}(0) = 0$ and for $x \in \mathbb{A} \setminus \{0\}$, $\mathit{sign}(x) = 1$ if $x > 0$, $\mathit{sign}(x) = -1$ if $x < 0$. The function that computes the sign is denoted $\texttt{Sign}(\mathbb{A}, d)$ with d, a representation of some item of $\mathbb{A}$. Since the procedures we describe may depend on additional properties like this one, we will indicate which properties are assumed for the algorithms.

Notations. The sign of a permutation that reverts the order of i items is denoted by $\varepsilon_i = (-1)^{\frac{i(i-1)}{2}}$. We denote by Rem the remainder of the Euclidean division in $\mathbb{A}[X]$: for polynomials $P, Q \in \mathbb{A}[X]$ with respective degrees p, q, $\mathit{Rem}(P, Q) \in \mathbb{F}_\mathbb{A}[X]$ is the unique polynomial of degree less than q such that there exists $C \in \mathbb{F}_\mathbb{A}[X]$ with $P = QC + \mathit{Rem}(P, Q)$.

Computing the degree of a gcd. We start with a characterization of the degree of the gcd of two polynomials that holds in any domain. The interest of this characterization is that it only involves whether some determinants in $\mathbb{A}$ are null and thus can be computed by additions, multiplications and zero-tests. Furthermore, subresultants will also be useful later on.

Definition 3.4 (Sylvester-Habicht matrices and subresultants). *Let $\mathbb{A}$ be a domain. Let $P, Q \in \mathbb{A}[X]$ with $P = \sum_{i \leq p} a_i X^i$ and $Q = \sum_{i \leq q} b_i X^i$ such that $a_p \neq 0$, $b_q \neq 0$ and $q \leq p$. Then the Sylvester-Habicht matrix of order j for $0 \leq j \leq \min(p-1, q)$ is the $(p+q-2j) \times (p+q-j)$ matrix $SyHa_j(P,Q)$ whose rows are $X^{q-j-1}P, \ldots, P, Q, \ldots, X^{p-j-1}Q$ considered as vectors with respect to the basis $X^{p+q-j-1}, \ldots, X, 1$.*
The j-th subresultant denoted $sRes_j(P,Q)$ is the determinant of the square matrix $SyHa_{j,j}(P,Q)$ obtained by taking the first $p+q-2j$ columns of $SyHa_j(P,Q)$. When $q < p$, this definition is extended for $q < j \leq p$ by: $sRes_p(P,Q) = a_p$, and $sRes_j(P,Q) = 0$ for $q < j \leq p-1$.

Remark 3.5. Observe that when $q < p$, $SyHa_q(P,Q)$ consists of $Q, \ldots, X^{p-q-1}Q$ (without any occurrence of P). Hence $sRes_q(P,Q)$ is the determinant of a matrix obtained by reverting the rows of $b_q Id_{p-q}$, which yields $sRes_q(P,Q) = \varepsilon_{p-q} b_q^{p-q}$.

Example 3.6. Consider polynomials $P = \alpha X^2 - 1$ and $Q = X + \beta$, obtained from the PoLITA of Fig. 1 when the value of X_1 has been fixed. By definition, we have $sRes_2(P,Q) = \alpha$, and by the above remark, $sRes_1(P,Q) = 1$. Precisely $SyHa_1(P,Q)$ is the one row matrix $(1, \beta)$ and $SyHa_{1,1}(P,Q) = (1)$. For $j = 0$, one must compute the determinant of the matrix whose rows are P, Q, XQ, namely

$$SyHa_0(P,Q) = \begin{pmatrix} \alpha & 0 & -1 \\ 0 & 1 & \beta \\ 1 & \beta & 0 \end{pmatrix}$$

whose determinant is $1 - \alpha\beta^2$.

Proposition 3.7. *Let $\mathbb{A}$ be a domain and $P, Q \in \mathbb{A}[X]$ with $P = \sum_{i \leq p} a_i X^i$ and $Q = \sum_{i \leq q} b_i X^i$ such that $a_p \neq 0$, $b_q \neq 0$ and $q \leq p$. Let $0 \leq j \leq \min(p-1, q)$. Then $deg(gcd(P,Q)) = j$ if and only if $sRes_0(P,Q) = \cdots = sRes_{j-1}(P,Q) = 0$ and $sRes_j(P,Q) \neq 0$. Consequently when $p = q$, $deg(gcd(P,Q)) = p$ if and only if $sRes_0(P,Q) = \cdots = sRes_{p-1}(P,Q) = 0$.*

Proof. Observe that $sRes_j(P,Q) = 0$ if and only if there is a non trivially null linear combination of polynomials $\alpha_{q-j-1}X^{q-j-1}P + \cdots + \alpha_0 P + \beta_0 Q + \cdots + \beta_{p-j-1}X^{p-j-1}Q$ of degree strictly less than j. This is equivalent to the existence of two non null polynomials $U = \sum_{i \leq q-j-1} \alpha_i X^i$ and $V = \sum_{i \leq p-j-1} \beta_i X^i$ such that $deg(UP + VQ) < j$.
We claim that $sRes_0(P,Q) = \cdots = sRes_{j-1}(P,Q) = 0$ if, and only if, it is the case that $deg(gcd(P,Q)) \geq j$, which will yield the desired conclusion. Assume that $deg(gcd(P,Q)) \geq j$ which is equivalent to $deg(lcm(P,Q)) \leq p+q-j$ which is equivalent to the existence of polynomials U, V with $deg(U) \leq q-j$, $deg(V) \leq p-j$ and $UP = -VQ$. Our previous observation implies that $sRes_0(P,Q) = \cdots = sRes_{j-1}(P,Q) = 0$.
The reverse implication is established by induction on j. When $sRes_0(P,Q) = 0$, the existence of U and V such that $UP + VQ = 0$ with $deg(U) < q$ and $deg(V) < p$ implies $deg(gcd(P,Q)) \geq 1$. When $sRes_0(P,Q) = \cdots = sRes_j(P,Q) = 0$,

the inductive hypothesis applied to $j-1$ implies $deg(gcd(P,Q)) \geq j$. From $sRes_j(P,Q)=0$, we again obtain U,V such that with $deg(U) < q-j$, $deg(V) < p-j$ and $deg(UP+VQ) < j$. Since $gcd(P,Q)$ divides $UP+VQ$ this implies that $UP+VQ=0$ and so $deg(lcm(P,Q)) < p+q-j$ and finally $deg(gcd(P,Q)) \geq j+1$. □

Due to the importance of the subresultant notion, we want a way to compute them efficiently. To this aim, we introduce the "polynomial" matrices and determinants. Let us introduce additional notations.

Definition 3.8. *Let $P_1,\ldots,P_m$ be polynomials in $\mathbb{A}[X]$of degrees less than n with $m \leq n$ and $P_i = \sum_{j<n} p_{i,j}X^j$. Then $pmat_n(P_1,\ldots,P_m)$ is the $m \times m$ matrix whose items are defined by:*

- *For all $i \leq m$, $j < m$, $pmat_n(P_1,\ldots,P_m)[i,j] = p_{i,n-j}$.*
- *For all $i \leq m$, $pmat_n(P_1,\ldots,P_m)[i,m] = P_i$.*

Additionally, let $pdet_n(P_1,\ldots,P_m) = det(pmat_n(P_1,\ldots,P_m))$.

Otherwise stated, the ith row of matrix $pmat_n(P_1,\ldots,P_m)$ consists of coefficients of P_i in descending order down to $n-m+1$ ended by polynomial P_i itself.

Definition 3.9. *Consider P,Q polynomials with respective degrees $p > q$. We define, for $0 \leq j \leq p$,*

- *for $0 \leq j \leq q$, $sResP_j(P,Q)pdet_{p+q-j}(X^{q-j-1}P,\ldots,P,Q,\ldots,X^{p-j-1}Q)$, that is $\det(SyHaP_j(P,Q))$, where $SyHaP_j(P,Q) = pmat_{p+q-j}(X^{q-j-1}P,\ldots,P,Q,\ldots,X^{p-j-1}Q)$.*
- *for $q < j < p-1$, $sResP_j(P,Q)=0$;*
- *for $j=p-1$, $sResP_j(P,Q)=Q$ (which is consistent with the original definition in case $q=p-1$);*
- *for $j=p$, $sResP_j(P,Q)=P$.*

From the above definition, one can straightforwardly see:

Proposition 3.10. *$sResP_j(P,Q)$ is a polynomial of degree at most j and the coefficient of degree j of this polynomial is $sRes_j(P,Q)$.*

Additional assumption. We assume here that the integral division is effective in $\mathbb{A}$: given $a,b \in \mathbb{A}$, there is an algorithm that answers whether there exists $c \in \mathbb{A}$ with $a=bc$ and returns c in the positive case. This is the case in particular in any ring over $\mathbb{Z}[X_1,\ldots,X_k] = \mathbb{Z}[X_1]\cdots[X_{k-1}][X_k]$ or $\mathbb{Q}[X_1,\ldots,X_k]$ where the algorithm consists in trying to perform a (recursive) Euclidean division, stopping and answering negatively when a coefficient of the quotient is not in the corresponding ring or there is a non null remainder. We denote the integral division by the usual fraction symbol since we will only use it when the result is defined.

Algorithm 2: Computing the subresultants for P, Q.

```
Subresultants($\mathbb{A}, P, p, Q, q$): a vector
Input: $P, Q$, non null polynomials in $\mathbb{A}[X]$ with respective degrees $p > q$
Output: the vector of subresultants $(sRes_i(P,Q))_{0 \leq i \leq p}$
Data: $SresP$ a vector over $\mathbb{A}[X]$ indexed by $[0, p]$
Data: $s, t$ vectors over $\mathbb{A}$ indexed by $[0, p]$; $i, j, k, \ell$, some indices
$SresP[p] \leftarrow P$; $s[p] \leftarrow 1$; $t[p] \leftarrow 1$; $SresP[p-1] \leftarrow Q$; $t[p-1] \leftarrow Q[q]$
if $q = p - 1$ then $s[p-1] \leftarrow t[p-1]$ else $s[p-1] \leftarrow 0$
for $\ell$ from $q+1$ to $p-2$ do $s[\ell] \leftarrow 0$
$SresP[q] \leftarrow \varepsilon_{p-q} t[p-1]^{p-q-1} Q$; $t[q] \leftarrow SresP[q][q]$; $s[q] \leftarrow t[q]$
$i \leftarrow p+1$; $j \leftarrow p$
while Degree($\mathbb{A}, SresP[j-1]$) $\neq -\infty$ do
    $k \leftarrow$ Degree($\mathbb{A}, SresP[j-1]$); $t[j-1] \leftarrow SresP[j-1][k]$
    if $k = j - 1$ then $s[j-1] \leftarrow t[j-1]$
    else
        for $\ell$ from $k+1$ to $j-1$ do $s[\ell] \leftarrow 0$
        $t[k] \leftarrow \varepsilon_{j-k} (\frac{t[j-1]}{s[j]})^{j-k-1} t[j-1]$; $s[k] \leftarrow t[k]$
        $SresP[k] \leftarrow \frac{s[k] sResP[j-1])}{t[j-1]}$
    end
    $SresP[k-1] \leftarrow \frac{-Rem(t[j-1]s[k]sResP[i-1], sResP[j-1])}{s[j]t[i-1]}$; $i \leftarrow j$; $j \leftarrow k$;
end
for $\ell$ from $0$ to $j-2$ do $s[\ell] \leftarrow 0$
return $s$
```

Our goal is to compute $sRes_j(P,Q)$ by decreasing values of j and only relying on Euclidean divisions that remain in $\mathbb{A}[X]$. For sake of clarity, we denote $s_j = sRes_j(P,Q)$ and t_j the leading coefficient of $sResP_j(P,Q)$ *except for* $s_p = t_p = 1$. When $sResP_j(P,Q)$ has degree j, we have $s_j = t_j$. Developing the last column w.r.t. the degrees of X and observing that for degrees $> j$ the corresponding vector of reals already occurs in a former column, we can safely substitute to the polynomials their truncation up to degree j. Then it is immediate that $sResP_j(P,Q) = 0$ iff there exist polynomials U,V with $deg(U) < q-j$, $deg(V) < p-j$ and $UP + VQ = 0$. As a consequence, for all $j' \leq j$, $sResP_{j'}(P,Q) = 0$.

The next proposition is the basis of Algorithm 2 for the efficient computation of subresultants. As can be deduced from this proposition, the computation consists in taking successive remainders of Euclidean divisions (up to some constant) in order to get $sResP_{i_j-1}(P,Q)$ and then some scalar multiplications and divisions to get $sResP_{i_{j+1}}(P,Q)$. Function `Degree` returns the degree of a polynomial in $\mathbb{N} \cup \{-\infty\}$ by looking at the first non null coefficient (using `Null` function).

Proposition 3.11. *Let P,Q be non null polynomials of $\mathbb{A}[X]$ with $p = deg(P) > deg(Q) = q$. There exists a sequence of strictly decreasing indices $i_1, i_2, \ldots, i_J$ with $i_1 = p+1$, $i_2 = p$, $i_3 = q$ that fulfills the following properties:*

- *for all $1 < j \leq J$, $sResP_{i_j}(P,Q)$ has degree i_j (and so $s_{i_j} = t_{i_j}$), for all $j < J$, $sResP_{i_j-1}(P,Q)$ has degree i_{j+1} and if $i_J > 0$ then for all $k < i_J$, $sResP_k(P,Q) = 0$ and $sResP_{i_{J-1}-1}(P,Q) = gcd(P,Q)$;*
- *for all $j < J$, when $i_j - 1 > i_{j+1}$, for all $i_{j+1} < k < i_j - 1$, $sResP_k(P,Q) = 0$ and $t_{i_j-1} sResP_{i_{j+1}}(P,Q) = s_{i_{j+1}} sResP_{i_j-1}(P,Q)$ with $s_{i_{j+1}} = \varepsilon_{i_j - i_{j+1}} \frac{(t_{i_j-1})^{i_j - i_{j+1}}}{(s_{i_j})^{i_j - i_{j+1} - 1}}$;*
- *for all $1 < j < J$, $s_{i_j} t_{i_{j-1}-1} sResP_{i_{j+1}-1}(P,Q) =$ $-Rem(s_{i_{j+1}} t_{i_j-1} sResP_{i_{j-1}-1}(P,Q), sResP_{i_j-1}(P,Q))$.*

Substituting in the equation of the third item $sResP_{i_{j-1}-1}(P,Q)$ by $\frac{t_{i_{j-1}-1}}{s_{i_j}} sResP_{i_j}(P,Q)$ (justified by the equation of the second item) and then multiplying by $\frac{t_{i_{j-1}-1}}{s_{i_j}}$ one also obtains:
$s_{i_j}^2 sResP_{i_{j+1}-1}(P,Q) = -Rem(s_{i_{j+1}} t_{i_j-1} sResP_{i_j}(P,Q), sResP_{i_j-1}(P,Q))$.

Proof. Let $R = Rem(P,Q)$. Let us look at $SyHaP_j(P,Q)$ for $j \leq q-1$. Write $C = \sum_{i \leq p-q} c_i X^i$ (the quotient of Euclidean division of P by Q). We have $R = P - \sum_{i \leq p-q} c_i X^i Q$. Due to this equality, changing the rows $X^{q-j-1}P, \ldots, P$ by $X^{q-j-1}R, \ldots, R$ does not modify the determinant $sResP_j(P,Q)$. We define the determinant D_j of the matrix obtained by reverting the order of the rows and replacing R by $-R$. The first operation amounts to multiplying by ε_{p+q-2j} and the second one by $(-1)^{q-j}$. Since $\varepsilon_{p+q-2j}(-1)^{q-j} = \varepsilon_{p-q}$, we have:
$D_j = \varepsilon_{p-q} sResP_j(P,Q)$.

We first prove the properties related to indexes between p and $q-1$. Let us look at the second item. For the first part by convention for all $q < j < p-1$,

$sResP_j(P,Q) = 0$. The second part of the second item corresponds to the case $j = 2$ with $s_p = a_p$, $t_{p-1} = s_q = b_q$. So the equation can be written as:

$$b_q sResP_q(P,Q) = s_q Q \text{ with } s_q = \varepsilon_{p-q}\frac{b_q^{p-q}}{1^{p-q-1}}$$

which is equivalent to:

$$sResP_q(P,Q) = \varepsilon_{p-q} b_q^{p-q-1} Q.$$

Since $sResP_q(P,Q) = pdet_p(Q,\dots,X^{p-q-1}Q)$, the result is immediate. Let us look at the third item: $D_{q-1} = -b^{p-q+1}R$. So

$$sResP_{q-1}(P,Q) = -Rem(\varepsilon_{p-q} b^{p-q+1} P, Q).$$

By convention, $s_p = t_p = 1$, $sResP_p(P,Q) = P$ and $sResP_{p-1}(P,Q) = Q$ implying $t_{p-1} = b_q$. Furthermore we have shown that $s_q = \varepsilon_{p-q} b^{p-q}$. Substituting in the previous equation establishes the third item.

We prove the remaining properties by induction on J. Let $R = Rem(P,Q) = 0$ which implies that $Q = gcd(P,Q)$ and $sResP_{q-1}(P,Q) = 0$. So the base case ($J = 3$) is established.

Let $R = Rem(P,Q) \neq 0$. Let r be the degree of R, we claim that:

$$\forall j < q-1 \; sResP_j(P,Q) = \varepsilon_{p-q} b_q^{p-r} sResP_j(Q,-R) \qquad (1)$$

When $j \leq r = deg(R)$, D_j can be obtained starting from $SyHaP_j(Q,-R)$ by adding the rows $X^{p-j-1}Q,\dots,X^{r-j}Q$ and taking the determinant. Thus $D_j = b_q^{p-r} sResP_j(Q,-R)$ and so $sResP_j(P,Q) = b_q^{p-r}\varepsilon_{p-q} sResP_j(Q,-R)$. When $r < j < q-1$ by definition $sResP_j(Q,-R) = 0$ but $sResP_j(P,Q) = D_j = 0$ since the polynomial matrix $pmat_{p+q-j}(X^{p-j-1}Q,\dots,Q,X^{q-j-1}R,\dots,R)$ is upper triangular up to its $p-j+1^{th}$ column and since the degree $X^{q-j-1}R$ is less than $q-1$, the diagonal term of this column is null.

Due to this proportionality between $sResP_j(P,Q)$ and $sResP_j(Q,-R)$ with factor $\varepsilon_{p-q} b_q^{p-r}$ and the inductive hypothesis, it only remains to prove that the two following equalities hold:

$$s_q t_{p-1} sResP_{r-1}(P,Q) = -Rem(s_r t_{q-1} sResP_{p-1}(P,Q), sResP_{q-1}(P,Q)) \quad (2)$$

and

$$s_r = \varepsilon_{q-r}\frac{(t_{q-1})^{q-r}}{(s_q)^{q-r-1}} \qquad (3)$$

For Equation (2), using the inductive hypothesis for the pair $(Q,-R)$, the following equation holds:

$$s_q'^2 sResP_{r-1}(Q,-R) = -Rem(s_r' t_{q-1}' sResP_q(Q,-R), sResP_{q-1}(Q,-R))$$

where the primed version of s_i and t_i are related to the pair $(Q,-R)$. By convention, $s_q' = 1$. So:

$$s_q t_{p-1} sResP_{r-1}(P,Q) = s_q t_{p-1}\varepsilon_{p-q} b_q^{p-r} sResP_{r-1}(Q,-R)$$

$$= (\varepsilon_{p-q} b_q^{p-q})(b_q)\varepsilon_{p-q} b_q^{p-r} sResP_{r-1}(Q, -R)$$

$$= -Rem((\varepsilon_{p-q} b_q^{p-r} s'_r)(\varepsilon_{p-q} b_q^{p-q+1} t'_{q-1}) sResP_q(Q, -R), sResP_{q-1}(Q, -R)).$$

Observe that the factor of proportionality established above implies that $s_r = \varepsilon_{p-q} b_q^{p-q+1} s'_r$.
Since $sResP_{q-1}(P, Q) = -\varepsilon_{p-q} b_q^{p-q+1} R$ and $sResP_{q-1}(Q, -R) = -R$, one obtains $t_{q-1} = \varepsilon_{p-q} b_q^{p-q+1}$. So:

$$s_q t_{p-1} sResP_{r-1}(P, Q) = -Rem(s_r t_{q-1} sResP_q(Q, -R), sResP_{q-1}(Q, -R))$$

$$= -Rem(s_r t_{q-1} Q, -R) = -Rem(s_r t_{q-1} Q, -\varepsilon_{p-q} b_q^{p-q+1} R)$$

$$= -Rem(s_r t_{q-1} sResP_{p-1}(P, Q), sResP_{q-1}(P, Q))$$

For Equation (3), let us look at the following matrices.

$$\begin{pmatrix} b_q & b_{q-1} & \ldots\ldots & X^{p-q-1}Q \\ 0 & b_q & \ldots\ldots & X^{p-q-2}Q \\ \ldots & \ldots & \ldots\ldots & \ldots \\ 0 & 0 & \ldots\ b_q & XQ \\ 0 & 0 & \ldots\ 0 & Q \end{pmatrix} \quad \begin{pmatrix} b_q & b_{q-1} & \ldots\ldots & X^{p-q}Q \\ 0 & b_q & \ldots\ldots & X^{p-q-1}Q \\ \ldots & \ldots & \ldots\ldots & \ldots \\ 0 & 0 & \ldots\ b_q & Q \\ 0 & 0 & \ldots\ 0 & -R \end{pmatrix}$$

The left matrix that we define D_q has been obtained by reverting the $p-q$ rows of $SyHaP_q(P, Q)$. So its determinant is equal to $\varepsilon_{p-q} sResP_q(P, Q)$. The right matrix is D_{q-1}. As we have already seen, its determinant is equal to $\varepsilon_{p-q} sResP_{q-1}(P, Q)$. Denoting $-R = \sum_{i \leq r} \alpha_i X^i$, it is now obvious that $b_q \alpha_r = \frac{t_{q-1}}{s_q}$. As a consequence, we obtain that:

$$sResP_{q-1}(P, Q) = -\varepsilon_{p-q} b_q^{p-q+1} R \tag{4}$$

Let us look at the following matrices.

$$\begin{pmatrix} b_q & b_{q-1} & \ldots\ldots\ldots & \ldots & \ldots\ldots & X^{p-r-1}Q \\ 0 & b_q & \ldots\ldots\ldots & \ldots & \ldots\ldots & X^{p-r-2}Q \\ \ldots & \ldots & \ldots\ldots\ldots & \ldots & \ldots\ldots & \ldots \\ 0 & 0 & \ldots\ b_q\ \ldots & \ldots & \ldots\ldots & Q \\ 0 & 0 & \ldots\ 0\ \ 0 & 0 & \ldots\ 0 & -R \\ 0 & 0 & \ldots\ 0 & & \ldots\ \alpha_r & -XR \\ \ldots & \ldots & \ldots\ldots\ldots & \ldots & \ldots\ldots & \ldots \\ 0 & 0 & \ldots\ldots\ 0 & \alpha_r & \ldots\ldots & -X^{q-r-2}R \\ 0 & 0 & \ldots\ldots\ \alpha_r & \alpha_{r-1} & \ldots\ldots & -X^{q-r-1}R \end{pmatrix} \begin{pmatrix} b_q & b_{q-1} & \ldots\ldots\ldots & \ldots & \ldots\ldots & X^{p-r-1}Q \\ 0 & b_q & \ldots\ldots\ldots & \ldots & \ldots\ldots & X^{p-r-2}Q \\ \ldots & \ldots & \ldots\ldots\ldots & \ldots & \ldots\ldots & \ldots \\ 0 & 0 & \ldots\ b_q\ \ldots & \ldots & \ldots\ldots & Q \\ 0 & 0 & \ldots\ldots\ \alpha_r & \alpha_{r-1} & \ldots\ldots & -X^{q-r-1}R \\ 0 & 0 & \ldots\ldots\ 0 & \alpha_r & \ldots\ldots & -X^{q-r-2}R \\ \ldots & \ldots & \ldots\ldots\ldots & \ldots & \ldots\ldots & \ldots \\ 0 & 0 & \ldots\ 0 & & \ldots\ \alpha_r & -XR \\ 0 & 0 & \ldots\ 0\ \ 0 & 0 & \ldots\ 0 & -R \end{pmatrix}$$

The left matrix is D_r and the right matrix has been obtained by reverting its last $q-r$ columns. So the determinant of the latter matrix is proportional to the determinant of the former with factor ε_{q-r}. On the other hand, the determinant of the right matrix is equal to the determinant of D_{j-1} multiplied by $(b_q \alpha_r)^{q-r-1}$. Combining the different equalities, we obtain that: $sRes_r(P, Q) = \varepsilon_{q-r}(\frac{t_{q-1}}{s_q})^{q-r-1} sRes_{q-1}(P, Q)$ and consequently $s_r = t_r = \varepsilon_{q-r} \frac{t_{q-1}^{q-r}}{s_q^{q-r-1}}$.

This concludes the proof. $\square$

Now we consider the special case of $\mathbb{A} = \mathbb{D}$, $\mathbb{D}$ being a sign-effective subring of $\mathbb{R}$. The main ingredient for analyzing real roots of a univariate polynomial is the Cauchy index. We denote by $Zer(P) = \{z \in \mathbb{R} \mid P(z) = 0\}$, $mult(P, z) = \max\{n \mid (X - z)^n | P\}$ and $Pole(Q/P) = \{z \in \mathbb{R} \mid mult(Q, z) < mult(P, z)\}$. For z in $Pole(Q/P)$, remark that $Q/P(w)$ goes to $+\infty$ or $-\infty$ as w tends to z on the right (respectively on the left), therefore the sign of Q/P keeps constant sufficiently close on the right (respectively on the left) of z.

Definition 3.12. *Let $P, Q \in \mathbb{D}[X]$. Then the Cauchy index of Q/P is defined by:*

$$Ind(Q/P) = \frac{1}{2} \sum_{z \in Pole(Q/P)} sign((Q/P)(z^+)) - sign((Q/P)(z^-))$$

where $sign((Q/P)(z^+))$ and $sign((Q/P)(z^-)))$ denote respectively the sign of the rational function Q/P at the right and at the left of z.

For $z \in Pole(Q/P)$, the value $sign((Q/P)(z^+)) - sign((Q/P)(z^-))$ in $\{-2, 0, 2\}$ depends on the parity of the difference $\mu_P - \mu_Q$ of respective multiplicities of z as root of P and Q, when $\mu_P \geq \mu_Q$ (and $\mu_Q = 0$ if z is not a root of Q).

Example 3.13. Recall polynomials $P = \alpha X^2 - 1$ and $Q = X + \beta$ of example 3.6. Let us compute the Cauchy index of Q/P for several values of α and β.

– Let P_1, Q_1 be the above polynomials with $\alpha = \sqrt{5}$ and $\beta = \frac{\sqrt{5}-7}{2}$. These values were obtained by setting X_1 to $\frac{1+\sqrt{5}}{2}$. The poles of Q_1/P_1 are $z_1 = -\frac{1}{\sqrt[4]{5}}$ and $z_2 = \frac{1}{\sqrt[4]{5}}$. One can see that $X + \beta$ remains negative between those poles. Hence

$$\begin{aligned} Ind(Q_1/P_1) &= \frac{1}{2}(sign(Q_1/P_1)(z_1^+) - sign(Q_1/P_1)(z_1^-) \\ &\qquad + sign(Q_1/P_1)(z_2^+) - sign(Q_1/P_1)(z_2^-)) \\ &= \frac{1}{2}(1 - (-1) + (-1) - 1) = 0. \end{aligned}$$

– Let P_2, Q_2 be the above polynomials with $\alpha = 2\sqrt{5} - 1$ and $\beta = 0$, which can be obtained by setting X_1 to $\sqrt{5}$. The poles of Q_2/P_2 are $z_1 = -\frac{1}{\sqrt{2\sqrt{5}-1}}$ and $z_2 = \frac{1}{\sqrt{2\sqrt{5}-1}}$. Now since Q_2 has a root between z_1 and z_2, hence

$$\begin{aligned} Ind(Q_2/P_2) &= \frac{1}{2}(sign(Q_2/P_2)(z_1^+) - sign(Q_2/P_2)(z_1^-) \\ &\qquad + sign(Q_2/P_2)(z_2^+) - sign(Q_2/P_2)(z_2^-)) \\ &= \frac{1}{2}(1 - (-1) + 1 - (-1)) = 2. \end{aligned}$$

The Cauchy index can be computed in several ways. First we observe that we can assume $q = deg(Q) < deg(P) = p$. Otherwise, let a_p be the leading coefficient of P and compute the Euclidean division of $a_p^{2\lceil \frac{q-p+1}{2} \rceil} Q$ by P:

$a_p^{2\lceil\frac{q-p+1}{2}\rceil}Q = PC + R$ with $deg(R) < deg(P)$. Then $Ind(Q/P) = Ind(R/P)$. The multiplication by an even power of a_p preserves the signs. Furthermore R is obtained by multiplications, additions and zero-tests so that it can be performed in a general domain $\mathbb{D}$ as indicated in Algorithm 3.

Algorithm 3: Computing a polynomial positively proportional to $Rem(Q, P)$

```
IntRem($\mathbb{D}, Q, q, P, p$): a polynomial with its degree
Input: $P \neq 0, Q$, polynomials in $\mathbb{D}[X]$ with respective degrees $p, q$
Output: a polynomial positively proportional to $Rem(Q, P)$
Data: $i, j$, some indices

if $q < p$ then return $Q, q$
for $i$ from $q - p$ downto $0$ do
    for $j$ from $0$ to $p - 1$ do $Q[i + j] \leftarrow P[p]Q[i + j] - P[j]Q[i + p]$
    for $j$ from $0$ to $i - 1$ do $Q[j] \leftarrow P[p]Q[j]$
end
for $i$ from $p$ to $q$ do $Q[i] \leftarrow 0$
if $q - p \mod 2 = 0$ then
    for $j$ from $0$ to $p - 1$ do $Q[j] \leftarrow P[p]Q[j]$
end
return $Q$, Degree($\mathbb{D}, Q$)
```

Here we use again the subresultants. Let $s = (s_p, \ldots, s_0)$ be a list of reals such that $s_p \neq 0$. Define s' as the shortest list such that $s = (s_p, 0, \ldots, 0) \cdot s'$. Then we inductively define:

$$PmV(s) = \begin{cases} 0 & \text{if } s' = \emptyset \\ PmV(s') + \varepsilon_{p-q} sign(s_p s_q) & \text{if } s' = (s_q, \ldots, s_0) \text{ and } p - q \text{ is odd} \\ PmV(s') & \text{otherwise} \end{cases}$$

Here acronym PmV means *(generalized) permanence minus variations* and as can be observed from the definition is related to the sign variations of the sequence s. An immediate property of the PmV is the following one. Let $x_p, \ldots, x_0$ be such that $sign(x_p) = \cdots = sign(x_0) \neq 0$, then $PmV(x_p s_p, \ldots, x_0 s_0) = PmV(s_p, \ldots, s_0)$.

Our approach consists in computing the PmV applied on subresultants.

Notations. If $p = deg(P) > q = deg(Q) \geq 0$, we denote by $sRes$ the tuple $(sRes_p, \ldots, sRes_0)$.

Example 3.14. For the polynomials of example 3.13, we have $sRes(P, Q) = (\alpha, 1, -\alpha\beta^2 + 1)$.

– In the first case, $sRes(P_1, Q_1) = (\sqrt{5}, 1, \frac{37-27\sqrt{5}}{2})$. Then

$$\begin{aligned}
PmV(sRes(P_1, Q_1)) &= PmV\left(1, \frac{37-27\sqrt{5}}{2}\right) + sign(\sqrt{5}) \\
&= PmV\left(\frac{37-27\sqrt{5}}{2}\right) + sign\left(\frac{37-27\sqrt{5}}{2}\right) + sign(\sqrt{5}) \\
&= 0 + sign\left(\frac{37-27\sqrt{5}}{2}\right) + sign(\sqrt{5}) = 0 + (-1) + 1 = 0.
\end{aligned}$$

– In the second case, $sRes(P_2, Q_2) = (2\sqrt{5} - 1, 1, 0)$. Then

$$\begin{aligned}
PmV(sRes(P_2, Q_2)) &= PmV(1, 1) + sign(2\sqrt{5} - 1) \\
&= PmV(1) + sign(1) + sign(2\sqrt{5} - 1) \\
&= 0 + 1 + 1 = 2.
\end{aligned}$$

Theorem 3.15. *Let $P, Q \in \mathbb{D}[X]$ with $p = deg(P) > q = deg(Q)$. Then:*
$$PmV(sRes(P, Q)) = Ind(Q/P)$$

Proof. Let $P = \sum_{i \le p} a_i X^i$, $Q = \sum_{i \le q} b_i X^i$ and let R be the remainder of the euclidean division of P by Q: $P = QC + R$. We consider two cases, according to whether $R = 0$ or not.

If $R = 0$ then $Q/P = 1/C$ with a_p/b_q the leading coefficient of C denoted by c_{p-q}, hence $sign(c_{p-q}) = sign(a_p b_q)$. Observe first that the sign of $1/C$ is unchanged between two consecutive poles. So the Cauchy index of $1/C$ will be half the sign of C at $+\infty$ minus the sign of C at $-\infty$. If $p - q$ is even then $C(x)$ will go to the same sign when x goes either to $+\infty$ or $-\infty$ entailing that $Ind(Q/P) = 0$. Otherwise it will go to opposite signs with the sign at $+\infty$ being $sign(a_p b_q)$, thus entailing that $Ind(Q/P) = sign(a_p b_q)$.

On the other hand, $sRes_p(P, Q) = a_p$, $sRes_j(P, Q) = 0$ for $q < j < p$ and $sRes_q(P, Q) = \varepsilon_{p-q} b_q^{p-q}$ from Remark 3.5. By Proposition 3.7, $sRes_j(P, Q) = 0$ for $j < q$. When $p - q$ is even, $PmV(sRes(P, Q)) = 0$ and when $p - q$ is odd, $PmV(sRes(P, Q)) = \varepsilon_{p-q} sign(a_p \varepsilon_{p-q} b_q^{p-q}) = sign(a_p b_q)$.

When $R \neq 0$, we claim that (1) $Ind(Q/P) = Ind(-R/Q) + sign(a_p b_q)$ when $p-q$ is odd and $Ind(Q/P) = Ind(-R/Q)$ otherwise and (2) $PmV(sRes(P, Q)) = PmV(sRes(Q, -R)) + sign(a_p b_q)$ when $p - q$ is odd and $PmV(sRes(P, Q)) = PmV(sRes(Q, -R))$ otherwise. This will imply the theorem by induction on the degree of P.

Let G be the gcd of P and Q and write $P = P_1 G$, $Q = Q_1 G$ and $R = R_1 G$. Obviously $Ind(Q/P) = Ind(Q_1/P_1)$ and $Ind(P/Q) = Ind(P_1/Q_1)$. In addition the signs of $PQ(x)$ and $P_1 Q_1(x)$ coincide on every point which is not a root of PQ. Since the roots of P_1 and Q_1 are distinct:

$$\frac{1}{2}(sign(PQ(+\infty)) - sign(PQ(-\infty))) = \frac{1}{2}(sign(P_1 Q_1(+\infty)) - sign(P_1 Q_1(-\infty)))$$

$$= \frac{1}{2} \sum_{z \in Zer(P_1Q_1)} sign((P_1Q_1)(z^+)) - sign((P_1Q_1)(z^-))$$
$$= \frac{1}{2} \sum_{z \in Zer(P_1)} sign((Q_1/P_1)(z^+)) - sign((Q_1/P_1)(z^-))$$
$$+\frac{1}{2} \sum_{z \in Zer(Q_1)} sign((P_1/Q_1)(z^+)) - sign((P_1/Q_1)(z^-))$$

$= Ind(Q_1/P_1)+Ind(P_1/Q_1) = Ind(Q/P)+Ind(P/Q) = Ind(Q/P)+Ind(R/Q)$.

Since $\frac{1}{2}(sign(PQ(\infty)) - sign(PQ(-\infty)))$ is null when $p-q$ is even and equal to $sign(a_pb_q)$ otherwise we obtain the first claim.

We recall Equation 1 where r is the degree of R:

$$\forall j < q-1 \ sResP_j(P,Q) = \varepsilon_{p-q} b_q^{p-r} sResP_j(Q,-R)$$

and Equation 4:

$$sResP_{q-1}(P,Q) = -\varepsilon_{p-q} b_q^{p-q+1} R.$$

Case 1: $q-1 > r$**.**

$Pmv(sRes(P,Q)) =$

$PmV(a_p, 0, \ldots, 0, \varepsilon_{p-q} b_q^{p-q}, 0, \ldots, 0, b_q^{p-r} \varepsilon_{p-q} sRes_r(Q,-R), \ldots, b_q^{p-r} \varepsilon_{p-q} sRes_0(Q,-R))$

Case 1.1: $q > r-1$ **and** $p-q$ **is even.**

$Pmv(sRes(P,Q)) =$

$PmV(\varepsilon_{p-q} b_q^{p-q}, 0, \ldots, 0, b_q^{p-r} \varepsilon_{p-q} sRes_r(Q,-R), \ldots, b_q^{p-r} \varepsilon_{p-q} sRes_0(Q,-R))$

$= PmV(b_q^{p-q}, 0, \ldots, 0, b_q^{p-r} sRes_r(Q,-R), \ldots, b_q^{p-r} sRes_0(Q,-R))$

$= PmV(1, 0, \ldots, 0, b_q^{q-r} sRes_r(Q,-R), \ldots, b_q^{q-r} sRes_0(Q,-R))$

$= PmV(b_q^{q-r}, 0, \ldots, 0, sRes_r(Q,-R), \ldots, sRes_0(Q,-R))$

Case 1.1.1: $q > r-1$ **and** $p-q$ **is even and** $q-r$ **is even.**

$= PmV(sRes_r(Q,-R), \ldots, sRes_0(Q,-R)) = PmV(sRes(Q,-R))$

Case 1.1.2: $q > r-1$ **and** $p-q$ **is even and** $q-r$ **is odd.**

$= PmV(b_q, 0, \ldots, 0, sRes_r(Q,-R), \ldots, sRes_0(Q,-R)) = PmV(sRes(Q,-R))$

Case 1.2: $q > r-1$ **and** $p-q$ **is odd.**

$Pmv(sRes(P,Q))$

$= PmV(\varepsilon_{p-q} b_q^{p-q}, 0, \ldots, 0, b_q^{p-r} \varepsilon_{p-q} sRes_r(Q,-R), \ldots, b_q^{p-r} \varepsilon_{p-q} sRes_0(Q,-R))$
$\quad +\varepsilon_{p-q} sign(a_p \varepsilon_{p-q} b_q^{p-q})$

$= PmV(b_q^{p-q}, 0, \ldots, 0, b_q^{p-r} sRes_r(Q,-R), \ldots, b_q^{p-r} sRes_0(Q,-R)) + sign(a_p b_q)$

$= PmV(b_q^{q-r}, 0, \ldots, 0, sRes_r(Q,-R), \ldots, sRes_0(Q,-R)) + sign(a_p b_q)$

where we conclude as in subcases 1.1.1 and 1.1.2.

Case 2: $q-1 = r$**.**

In this case using Equation 4, $sRes_{q-1}(P,Q) = -\varepsilon_{p-q} b - q^{p-q+1} c_r = \varepsilon_{p-q} b_q^{p-r}(-c-r)$ where c_r is the leading coefficient of R

So $Pmv(sRes(P,Q))$

$= PmV(a_p, 0, \ldots, 0, \varepsilon_{p-q} b_q^{p-q}, b_q^{p-r} \varepsilon_{p-q} sRes_{q-1}(Q,-R), \ldots, b_q^{p-r} \varepsilon_{p-q} sRes_0(Q,-R))$

And we conclude as in case 1.

□

Algorithm 4 describes how to compute the PmV and so the Cauchy index of two polynomials. Now let us introduce the Tarski query.

Algorithm 4: Computing the generalized permanences minus variations

```
PmVPol($\mathbb{A}, P, p, Q, q$): an integer
Input: $P, Q$, polynomials $\mathbb{A}[X]$ of degree $p$ and $q$ with $q < p$
Output: $PmV(sRes_p(P,Q), \ldots, sRes_0(P,Q))$
Data: $j$ an index, $s_p, \ldots, s_0$ a sequence of signs
Data: $sReS(P,Q)$ a sequence of items of $\mathbb{A}$

if $q = -\infty$ then return $0$
// consistently with Cauchy index definition
$sRes(P,Q) \leftarrow$ SubResultants($\mathbb{A}, P, p, Q, q$)
// The subresultants computation depends on $\mathbb{A}$
// since Algorithm 2 has an additional assumption.
for $j$ from $0$ to $p$ do $s_j \leftarrow$ Sign($\mathbb{A}, sRes_j(P,Q)$)
return $PmV(s_p, \ldots, s_0)$ // by applying the definition
```

Definition 3.16 (Tarski query). *Let $P, Q \in \mathbb{D}[X]$. Then:*

$$TaQ(Q,P) = \sum_{z \in Zer(P)} sign(Q(z)).$$

The Tarski query is closely related to the Cauchy index as established by the next proposition.

Proposition 3.17. *Let $P, Q \in \mathbb{D}[X]$. Then:*

$$TaQ(Q,P) = Ind(P'Q/P).$$

Proof. Let z be a root of P with multiplicity μ. Then $P'Q/P = Q(\frac{\mu}{X-z} + R)$ with R a rational function with no pole at z. If $Q(z) = 0$ then $P'Q/P$ has no pole in z. Otherwise $sign((P'Q/P)(z^+)) = sign(Q(z))$ and $sign((P'Q/P)(z^-)) = -sign(Q(z))$. The assertion of the proposition follows. □

Example 3.18.
- For $P_1 = \sqrt{5}X^2 + 1$ and $Q_1 = X + \frac{\sqrt{5}-7}{2}$, we have $P_1' = 2\sqrt{5}X$. The sign of $P_1'Q_1$ around the poles of $P_1'Q_1/P_1$ is constant: positive around z_1 and negative around z_2. Hence $Ind(P_1'Q_1/P_1) = \frac{1}{2}(-1-1+(-1)-1) = -2$. On the other hand, since the sign of Q_1 is negative at both z_1 and z_2, $TaQ(Q_1, P_1) = -1 + (-1) = -2$.
- For $P_2 = (2\sqrt{5}-1)X^2 - 1$ and $Q_2 = X$, we have $P_2' = (4\sqrt{5}-2)X$. The sign of $P_2'Q_2$ is always non-negative, hence it is so at the poles of $P_2'Q_2/P_2$, where it is non-zero. Hence $Ind(P_2'Q_2/P_2) = \frac{1}{2}(-1-1+1-(-1)) = 0$ while Q_2 has the same sign as the roots of P_2, so $TaQ(Q_2, P_2) = -1 + 1 = 0$.

In fact the Tarski question is an auxiliary value. The values we are really interested in are the following counters:

- $\mathbf{nb}_P(Q)[-1] = |\{z \in Zer(P) \mid Q(z) < 0\}|$;
- $\mathbf{nb}_P(Q)[0] = |\{z \in Zer(P) \mid Q(z) = 0\}|$.

– $\mathbf{nb}_P(Q)[1] = |\{z \in Zer(P) \mid Q(z) > 0\}|$;

The following lemma whose proof is obvious is the key for computing such counters.

Lemma 3.19. *The Tarski queries and root counters are related by:*

– $TaQ(1, P) = \mathbf{nb}_P(Q)[-1] + \mathbf{nb}_P(Q)[0] + \mathbf{nb}_P(Q)[1]$*;*
– $TaQ(Q, P) = -\mathbf{nb}_P(Q)[-1] + \mathbf{nb}_P(Q)[1]$*;*
– $TaQ(Q^2, P) = \mathbf{nb}_P(Q)[-1] + \mathbf{nb}_P(Q)[1]$*.*

Example 3.20. We previously computed $TaQ(Q_1, P_1) = -2$ (see Example 3.18). The value $TaQ(1, P_1)$, actually computed through $Ind(P_1'/P_1)$ yields the number of roots of P_1, which is 2. Finally, computing $TaQ(Q_1^2, P_1)$ can also be done through the Cauchy index, and yields the number of roots of P_1 that are not roots of Q_1, in this case also 2.

As a result, solving the system induced by the above lemma, there are two roots of P_1 where Q_1 is strictly negative, and no root of P_1 where Q_1 is positive or null. The polynomial Q_1 has degree 1, this shows that both roots of P_1 are strictly smaller than the (only) root of Q_1.

Thus defining the invertible matrix $\mathbf{M}_1$ and vector $\mathbf{TaQ}_P(Q)$ by:

$$\mathbf{M}_1 = \begin{pmatrix} 1 & 1 & 1 \\ -1 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix} \quad \mathbf{TaQ}_P(Q) = \begin{pmatrix} \mathbf{TaQ}_P(Q)[0] \\ \mathbf{TaQ}_P(Q)[1] \\ \mathbf{TaQ}_P(Q)[2] \end{pmatrix} = \begin{pmatrix} TaQ(Q^0, P) \\ TaQ(Q^1, P) \\ TaQ(Q^2, P) \end{pmatrix}$$

we obtain:

Proposition 3.21.

$$\mathbf{TaQ}_P(Q) = \mathbf{M}_1 \cdot \mathbf{nb}_P(Q)$$

As we are interested in determining the simultaneous signs of polynomials evaluated on the roots of another polynomial we generalize mappings $\mathbf{nb}_P$ and $\mathbf{TaQ}_P$ to a sequence of polynomials.

Definition 3.22 (Generalized counters and Tarski queries). *Let* $P \in \mathbb{D}[X]$ *and* $\mathcal{Q} = (Q_1, \ldots, Q_m)$ *be a finite sequence of* $\mathbb{D}[X]$*. Then:*
$\mathbf{nb}_P(\mathcal{Q})$ *is an integer vector whose support is* $\{-1, 0, 1\}^{\{1,\ldots,m\}}$ *such that:*

$$\mathbf{nb}_P(\mathcal{Q})[i_1, \ldots, i_m] = |\{z \in Zer(P) \mid \forall j \leq m \; sign(Q_j(z)) = i_j\}|$$

$\mathbf{TaQ}_P(\mathcal{Q})$ *is an integer vector whose support is* $\{0, 1, 2\}^{\{1,\ldots,m\}}$ *such that:*

$$\mathbf{TaQ}_P(\mathcal{Q})[i_1, \ldots, i_m] = TaQ(Q_1^{i_1} \cdots Q_m^{i_m})$$

The tensor product of two matrices $\mathbf{A}$ of dimension $m_a \times n_a$ and $\mathbf{B}$ of dimension $m_b \times n_b$ is the matrix $\mathbf{A} \otimes \mathbf{B}$ of dimension $m_a m_b \times n_a n_b$ defined by: $\mathbf{A} \otimes \mathbf{B}[(i_a, i_b), (j_a, j_b)] = \mathbf{A}[i_a, j_a]\mathbf{B}[i_b, j_b]$. We inductively define for $t > 1$, $\mathbf{M}_t = \mathbf{M}_1 \otimes \mathbf{M}_{t-1}$.

Proposition 3.23. *Let $P \in \mathbb{D}[X]$ and $\mathcal{Q} = (Q_1, \dots, Q_m)$ a finite sequence of $\mathbb{D}[X]$. Then:*

$$\mathbf{TaQ}_P(\mathcal{Q}) = \mathbf{M}_m \cdot \mathbf{nb}_P(\mathcal{Q}).$$

Proof. Observe that both $\mathbf{TaQ}_P(\mathcal{Q})$ and $\mathbf{nb}_P(\mathcal{Q})$ only depend on $Zer(P)$. Thus w.l.o.g we assume that $P = \prod_i X - z_i$ with all z_i distinct. In this case,

$$\mathbf{TaQ}_P(\mathcal{Q}) = \sum_i \mathbf{TaQ}_{X-z_i}(\mathcal{Q}) \text{ and } \mathbf{nb}_P(\mathcal{Q}) = \sum_i \mathbf{nb}_{X-z_i}(\mathcal{Q}).$$

So we are left with the case $P = X - z$. For all $(i_1, \dots, i_m)$,

$$\begin{aligned} \mathbf{TaQ}_P(\mathcal{Q})[i_1, \dots, i_m] &= TaQ_P(Q_1^{i_1} \cdots Q_m^{i_m}) \\ &= sign(Q_1^{i_1}(z) \cdots Q_m^{i_m}(z)) \\ &= \prod_j sign(Q_j^{i_j}(z)) \ = \ \prod_j TaQ_P(Q_j^{i_j}) \end{aligned}$$

Therefore by definition of tensor product,
$\mathbf{TaQ}_P(\mathcal{Q}) = \mathbf{TaQ}_P(Q_1) \otimes \cdots \otimes \mathbf{TaQ}_P(Q_m)$.
On the other hand, for all $(i_1, \dots, i_m)$, $\mathbf{nb}_P(\mathcal{Q})[i_1, \dots, i_m] = \mathbf{1}_{\bigwedge_j sign(Q_j(z))=i_j}$
$= \prod_j \mathbf{1}_{sign(Q_j(z))=i_j} = \prod_j \mathbf{nb}_P(\mathcal{Q}_j)[i_j]$. Therefore,
$\mathbf{nb}_P(\mathcal{Q}) = \mathbf{nb}_P(Q_1) \otimes \cdots \otimes \mathbf{nb}_P(Q_m)$.
So $\mathbf{TaQ}_P(\mathcal{Q}) = \mathbf{TaQ}_P(Q_1) \otimes \cdots \otimes \mathbf{TaQ}_P(Q_m)$
$= \mathbf{M_1} \cdot \mathbf{nb}_P(Q_1) \otimes \cdots \otimes \mathbf{M_1} \cdot \mathbf{nb}_P(Q_m)$ *using Proposition 3.21*
$= (\mathbf{M_1} \otimes \cdots \otimes \mathbf{M_1}) \cdot (\mathbf{nb}_P(Q_1) \otimes \cdots \otimes \mathbf{nb}_P(Q_m))$ *using a property of tensor product*
$= \mathbf{M_m} \cdot \mathbf{nb}_P(\mathcal{Q})$.

□

Using elementary properties of the tensorial product, one gets the following corollary.

Corollary 3.24. *Let $P \in \mathbb{D}[X]$ and $\mathcal{Q} = (Q_1, \dots, Q_m)$ a finite sequence of $\mathbb{D}[X]$. Then:*

$$\mathbf{nb}_P(\mathcal{Q}) = (\mathbf{M}_m)^{-1} \cdot \mathbf{TaQ}_P(\mathcal{Q}) = \left((\mathbf{M}_1)^{-1} \otimes \cdots \otimes (\mathbf{M}_1)^{-1}\right) \cdot \mathbf{TaQ}_P(\mathcal{Q}).$$

While the previous corollary provides a way to compute the number of zeroes of P per sign realization at family $\mathcal{Q}$, the procedure is highly inefficient w.r.t m. Indeed M_m has size $3^m \times 3^m$ while the values and the size of the support of vector $\mathbf{nb}_P(\mathcal{Q})$ remain bounded by the number of zeroes of P. So in the next paragraphs, we refine the procedure by iteratively computing $\mathbf{nb}_P(Q_i, \dots, Q_m)$ by decreasing values of i and using the intermediate result to reduce the size of the matrix to be inverted at the next computation step.

Definition 3.25. *Let m be an integer, $\Sigma \subseteq \{-1, 0, 1\}^m$ and $A \subseteq \{0, 1, 2\}^m$. Then A is* adapted *to Σ if the (sub)matrix $M_m[A, \Sigma]$ is invertible.*

Since M_m is invertible any Σ admits some A. However we need a way to efficiently compute such an A.

Definition 3.26. *Let $\Sigma \subseteq \{-1,0,1\}^m$. Then $A(\Sigma)$ is inductively defined by:*

– *If $m = 1$ then:*
  1. *When $|\Sigma| = 1$, $A(\Sigma) = \{0\}$*
  2. *When $|\Sigma| = 2$, $A(\Sigma) = \{0,1\}$*
  3. *When $|\Sigma| = 3$, $A(\Sigma) = \{0,1,2\}$*

– *Let $\Sigma \subseteq \{-1,0,1\}^{m+1}$.*
  For $k \in \{1,2,3\}$, define $\Sigma_k = \{\sigma \in \{-1,0,1\}^m \mid |\{(i,\sigma) \in \Sigma\}| \geq k\}$.
  Then $A(\Sigma) = \{0\} \times A(\Sigma_1) \cup \{1\} \times A(\Sigma_2) \cup \{2\} \times A(\Sigma_3)$.

Observe that $\Sigma_3 \subseteq \Sigma_2 \subseteq \Sigma_1$ and that $|\Sigma_3| + |\Sigma_2| + |\Sigma_1| = |\Sigma|$.

Proposition 3.27. *Let $\Sigma \subseteq \{-1,0,1\}^m$. Then $A(\Sigma)$ is adapted to Σ.*

Proof. The base case $m = 1$ is established by a straightforward examination of M_1. Assume that the result holds for m and consider $\Sigma \subseteq \{-1,0,1\}^{m+1}$. For $\sigma \in \Sigma_1$, we denote by C_σ the column of matrix $M_m[\{0,1,2\}^m, \Sigma_1]$ indexed by σ. Then columns of matrix $M_{m+1}[\{0,1,2\}^{m+1}, \Sigma]$ are:

- $C_{(-1,\sigma)} = \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix} \otimes C_\sigma = \begin{pmatrix} C_\sigma \\ -C_\sigma \\ C_\sigma \end{pmatrix}$ if $(-1,\sigma) \in \Sigma$,

- $C_{(0,\sigma)} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \otimes C_\sigma = \begin{pmatrix} C_\sigma \\ 0 \\ 0 \end{pmatrix}$ if $(0,\sigma) \in \Sigma$,

- $C_{(1,\sigma)} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \otimes C_\sigma = \begin{pmatrix} C_\sigma \\ C_\sigma \\ C_\sigma \end{pmatrix}$ if $(1,\sigma) \in \Sigma$.

For $\sigma \in \Sigma_1$, we pick a minimal $k_{\sigma,1}$ such that $(k_{\sigma,1},\sigma) \in \Sigma$. For $\sigma \in \Sigma_2$, we pick a minimal $k_{\sigma,2} \neq k_{\sigma,1}$ such that $(k_{\sigma,2},\sigma) \in \Sigma$. For $\sigma \in \Sigma_3$, we pick the unique $1 = k_{\sigma,3} \notin \{k_{\sigma,1}, k_{\sigma,2}\}$ such that $(k_{\sigma,3},\sigma) \in \Sigma$. Let us reorder the columns of matrix $M_{m+1}[\{0,1,2\}^{m+1}, \Sigma]$ as follows. The first $|\Sigma_1|$ columns are those indexed by all $(k_{\sigma,1},\sigma) \in \Sigma$. The next $|\Sigma_2|$ columns are those indexed by all $(k_{\sigma,2},\sigma) \in \Sigma$. The last $|\Sigma_3|$ columns are those indexed by all $(k_{\sigma,3},\sigma) \in \Sigma$. We then perform on this matrix some columns operations that let the linear independence status of rows unchanged:

– when $k_{\sigma,1} = -1$ and $k_{\sigma,2} = 0$ then $C_{0,\sigma} \leftarrow C_{0,\sigma} - C_{-1,\sigma}$ so that
  $C_{0,\sigma} = \begin{pmatrix} 0 \\ C_\sigma \\ -C_\sigma \end{pmatrix}$.

– when $k_{\sigma,1} = -1$ and $k_{\sigma,2} = 1$ then $C_{1,\sigma} \leftarrow \frac{1}{2}(C_{1,\sigma} - C_{-1,\sigma})$ so that
  $C_{1,\sigma} = \begin{pmatrix} 0 \\ C_\sigma \\ 0 \end{pmatrix}$.

– when $k_{\sigma,1} = 0$ and $k_{\sigma,2} = 1$ then $C_{1,\sigma} \leftarrow C_{1,\sigma} - C_{-1,\sigma}$ so that

$$C_{1,\sigma} = \begin{pmatrix} 0 \\ C_\sigma \\ C_\sigma \end{pmatrix}.$$

– when $k_{\sigma,3}$ is defined (and so equal to 1) then $C_{1,\sigma} \leftarrow \frac{1}{2}(C_{1,\sigma} - 2C_{0,\sigma} + C_{-1,\sigma})$ so that

$$C_{1,\sigma} = \begin{pmatrix} 0 \\ 0 \\ C_\sigma \end{pmatrix}.$$

The resulting matrix has a triangular form :

$$\begin{pmatrix} M_m[\{0,1,2\}^m, \Sigma_1] & 0 & 0 \\ * & M_m[\{0,1,2\}^m, \Sigma_2] & 0 \\ * & * & M_m[\{0,1,2\}^m, \Sigma_3] \end{pmatrix}$$

Due to this triangular form, the first $|\Sigma_1| + |\Sigma_2| + |\Sigma_3|$ independent rows of $M_{m+1}[\{0,1,2\}^{m+1}, \Sigma]$ are the first $|\Sigma_1|$ rows of the first diagonal block followed by the first $|\Sigma_2|$ rows of the second diagonal block and the first $|\Sigma_3|$ rows of the third diagonal block. □

Computing inductively $A(\Sigma)$ seems to require three "recursive calls". However observing that $\Sigma_3 \subseteq \Sigma_2 \subseteq \Sigma_1$ and using the next proposition we will obtain an efficient computation.

Proposition 3.28. *Let $\Sigma' \subseteq \Sigma \subseteq \{-1,0,1\}^m$. Then $A(\Sigma')$ is obtained by extracting the first $|\Sigma'|$ linearly independent rows of matrix $M_m[A(\Sigma), \Sigma']$.*

Proof. We proceed by induction on m. The base case $m = 1$ is an immediate consequence of the definition of $A(\Sigma)$.
Assume that result holds for m and consider $\Sigma' \subseteq \Sigma \subseteq \{-1,0,1\}^{m+1}$. Define as in Definition 3.26, Σ'_1, Σ'_2 and Σ'_3. Observe that for all i, $\Sigma'_i \subseteq \Sigma_i$. Consider matrix $M_{m+1}[\{-1,0,1\}^{m+1}, \Sigma']$. After performing the same linear transformations on the columns as those of the previous proof, we obtain the following matrix:

$$\begin{pmatrix} M_m[\{0,1,2\}^m, \Sigma'_1] & 0 & 0 \\ * & M_m[\{0,1,2\}^m, \Sigma'_2] & 0 \\ * & * & M_m[\{0,1,2\}^m, \Sigma'_3] \end{pmatrix}$$

Thus the first maximal set of independent rows of this matrix will be obtained by the first maximal sets of independent rows in the three diagonal blocks. Applying the induction hypothesis, this corresponds to the first maximal set of independent rows of the following matrix:

$$\begin{pmatrix} M_m[A(\Sigma_1), \Sigma'_1] & 0 & 0 \\ * & M_m[A(\Sigma_2), \Sigma'_2] & 0 \\ * & * & M_m[A(\Sigma_3), \Sigma'_3] \end{pmatrix}$$

which (by the inverse linear transformations) is equivalent to looking for the first $|\Sigma'|$ linearly independent rows of matrix $M_m[A(\Sigma), \Sigma']$.

□

Algorithm 5 implements the whole method developped above.

Algorithm 5: Computing sign realizations of family $\mathcal{Q}$ at roots of P

```
SignRealization($\mathbb{D}, P, p, \mathcal{Q}$): a non null vector with its support
Input: $P$, a non null polynomial in $\mathbb{D}[X]$ with degree $p$
Input: $\mathcal{Q} = \{(Q_1, q_1), \ldots, (Q_m, q_m)\}$, a family of non null polynomials in $\mathbb{D}[X]$
Output: the vector counting the sign realizations for $\mathcal{Q}$ by the roots of $P$
Data: $e_1, \ldots, e_m$ degrees in $\{0, 1, 2\}$
Data: $\mathbf{TaQ}$ a vector indexed by vectors of degrees
Data: $\mathbf{nb}$ a vector indexed by vectors of signs
Data: $R$, a polynomial in $\mathbb{D}[X]$, $r$ a degree
Data: $\mathbf{M}$, an integer matrix
Data: $extA, A, A_1, A_2, A_3$, sets of vectors of degrees
Data: $ext\Sigma, \Sigma, \Sigma_1, \Sigma_2, \Sigma_3$, sets of vectors of signs
for $e_m$ in $\{0, 1, 2\}$ do
    $R \leftarrow P'Q_m^{e_m}$; $(R, r) \leftarrow$ IntRem($\mathbb{D}, R, e_m q_m + p - 1, P, p$)
    // see Algorithm 3 for IntRem
    $\mathbf{TaQ}[e_m] \leftarrow$ PmVPol($\mathbb{D}, P, p, R, r$)
end
$\mathbf{nb} \leftarrow \mathbf{M}_1^{-1} \cdot \mathbf{TaQ}$
if $\mathbf{nb} = \mathbf{0}$ then return $\emptyset, -$ // $P$ has no roots
$\Sigma \leftarrow supp(\mathbf{nb})$
if $|\Sigma| = 1$ then $A \leftarrow \{0\}$
else if $|\Sigma| = 2$ then $A \leftarrow \{0, 1\}$
else $A \leftarrow \{0, 1, 2\}$
$\mathbf{nb} \leftarrow \mathbf{nb}_{|\Sigma}$; $\mathbf{M} \leftarrow \mathbf{M}_{1|A \times \Sigma}$
for $i$ from $m - 1$ downto $1$ do
    $ext\Sigma \leftarrow \{-1, 0, 1\} \times \Sigma$; $extA \leftarrow \{0, 1, 2\} \times A$; $\mathbf{extM} \leftarrow \mathbf{M}_1 \otimes \mathbf{M}$
    for $(e_i, \ldots, e_m)$ in $extA$ do
        $R \leftarrow P' \prod_{i \le j \le m} Q_j^{e_j}$; $(R, r) \leftarrow$ IntRem($\mathbb{D}, R, \sum_{i \le j \le m} q_j e_j, P, p$)
        $\mathbf{TaQ}[(e_i, \ldots, e_m)] \leftarrow$ PmVPol($\mathbb{D}, P, p, R, r$)
    end
    $\mathbf{nb} \leftarrow \mathbf{extM}^{-1} \cdot \mathbf{TaQ}$
    $\Sigma_1 \leftarrow \Sigma$
    $\Sigma_2 \leftarrow \{\sigma \in \Sigma \mid |\{(i, \sigma) \in supp(\mathbf{nb})\}| \ge 2\}$;
    $\Sigma_3 \leftarrow \{\sigma \in \Sigma \mid |\{(i, \sigma) \in supp(\mathbf{nb})\}| \ge 3\}$
    $A_1 \leftarrow A$
    $A_2 \leftarrow$ the indexes of the first $|\Sigma_2|$ linearly independent rows of $\mathbf{M}_{|A \times \Sigma_2}$
    $A_3 \leftarrow$ the indexes of the first $|\Sigma_3|$ linearly independent rows of $\mathbf{M}_{|A_2 \times \Sigma_3}$
    $\Sigma \leftarrow supp(\mathbf{nb})$
    $A \leftarrow \{0\} \times A_1 \cup \{1\} \times A_2 \cup \{2\} \times A_3$
    $\mathbf{nb} \leftarrow \mathbf{nb}_{|\Sigma}$; $\mathbf{M} \leftarrow \mathbf{extM}_{|A \times \Sigma}$
end
return $\Sigma, \mathbf{nb}$
```

Defining and computing encodings for roots

Definition 3.29 (Thom-encoding). *Let $P \in \mathbb{D}[X]$ with $deg(P) = p > 0$ and $x \in \mathbb{R}$. The P-encoding of x is the vector:*

$$\sigma_P(x) = (sign(P(x)), sign(P'(x)), \ldots, sign(P^{(p)}(x))).$$

A P-code is a vector of signs indexed by $\{0, \ldots, deg(P)\}$.

Proposition 3.30. *Let $P \in \mathbb{D}[X]$ and σ be a P-code. Then:*

– *$\sigma_P^{-1}(\sigma)$ is either empty, a point or an open interval.*
– *Let $x \neq x'$ be two roots of P. Then $\sigma_P(x) \neq \sigma_P(x')$.*
– *Let x, x' with $\sigma_P(x) \neq \sigma_P(x')$. Then $x < x'$ if and only if, denoting k the largest index with $\sigma_P(x)[k] \neq \sigma_P(x')[k]$:*
  1. *either $\sigma_P(x)[k+1] = 1$ and $\sigma_P(x)[k] < \sigma_P(x')[k]$;*
  2. *or $\sigma_P(x)[k+1] = -1$ and $\sigma_P(x)[k] > \sigma_P(x')[k]$.*

Proof. We proceed by induction on the degree of P. The case $deg(P) = 1$ is obvious. Assume that it is valid for all P such that $deg(P) \leq i$. Consider P with $deg(P) = i + 1$. Apply the inductive hypothesis on σ restricted to its i last components, denoted σ', and on P'. When $\sigma_{P'}^{-1}(\sigma')$ is empty or a point then the result is immediate. When $\sigma_{P'}^{-1}(\sigma')$ is an interval, then $\sigma[1] \neq 0$. Thus $P(x)$ is a strictly monotonous function on the interval which meets 0 at most once. This implies the result.

The second assertion is a direct consequence of the first assertion.

Considering the third assertion, $\sigma_{P^{(k+1)}}(x) = \sigma_{P^{(k+1)}}(x')$. Since $x \neq x'$, the second assertion implies that $\sigma_{P^{(k+1)}}(x) \neq 0$.
Since $P^{(k+1)}$ is constant in $[\min(x, x'), \max(x, x')]$, this implies the third assertion.

□

Algorithm 6: Computing the Q-encoding of roots of P

```
RootCoding($\mathbb{D}, P, p, Q, q$): a list
Input: $P, Q$, non null polynomials in $\mathbb{D}[X]$ with respective degrees $p, q$
Output: a list of the $Q$-encoding of roots of $P$
Data: $(s_0, \ldots, s_q)$ a vector of signs
Data: nb a vector indexed by vectors of signs
Data: $\Sigma$ a set of vectors of signs
$(\Sigma, \mathbf{nb}) \leftarrow$ SignRealization($\mathbb{D}, P, p, \{(Q^{(0)}, q), \ldots, (Q^{(q)}, 0)\}$)
Order the $Q$-encodings $(s_0, \ldots, s_q)$ of the support $\Sigma$ of nb
using Proposition 3.30 and duplicating them w.r.t. $\mathbf{nb}[(s_0, \ldots, s_q)]$
return this list of encodings
```

Example 3.31. Let us consider the P_1-encoding of reals for $P_1 = \sqrt{5}X^2 - 1$. First remark that the second derivative is always positive, hence the third component of the P_1-encoding of any real number is always $+1$. This encoding divides the real line into seven intervals:

- $]-\infty, -\frac{1}{\sqrt[4]{5}}[$ is encoded into $(+1, -1, +1)$, since for x in this interval, $P(x)$ is positive but decreasing.
- The first root $[-\frac{1}{\sqrt[4]{5}}, -\frac{1}{\sqrt[4]{5}}]$ is encoded into $(0, -1, +1)$.
- $]-\frac{1}{\sqrt[4]{5}}, 0[$ corresponds to $(-1, -1, +1)$.
- The point $[0, 0]$ is encoded by $(-1, 0, +1)$.
- $]0, \frac{1}{\sqrt[4]{5}}[$ corresponds to $(-1, +1, +1)$.
- The second root $[\frac{1}{\sqrt[4]{5}}, \frac{1}{\sqrt[4]{5}}]$ is encoded into $(0, +1, +1)$.
- $]\frac{1}{\sqrt[4]{5}}, +\infty[$ is encoded into $(+1, +1, +1)$.

As a consequence of our previous developments, we are now in position to perform two main computations in $\mathbb{D}[X]$: (1) determining the number of roots of a polynomial P and computing their P-encoding, and (2) computing the Q-encoding of roots of a polynomial P. Both results are obtained by Algorithm 6. For the first goal it is sufficient to call `PmVPol`(P, P') and if the result is non null to call `RootCoding`(P, P).

3.2 Triangular systems

While we only stated the effective properties of (a representation of) $\mathbb{D}$ in the previous parts, we now consider specific representations of real subrings of the form $\mathbb{D} = \mathbb{Q}[\alpha_1, \ldots, \alpha_\ell]$ where the α_i's are real algebraic numbers. Such representations are called triangular systems and we will show (in Proposition 3.34) that they are sign-effective. In the sequel, the leading coefficient of $P = \sum_{i \leq p} a_i X^i$ in $\mathbb{D}[X]$ with $deg(P) = p$ is denoted $lcof(P) = a_p$. Note that the leading coefficient of a polynomial P in $\mathbb{Q}[X_1, \ldots, X_{i-1}][X_i]$ is itself a polynomial in $\mathbb{Q}[X_1, \ldots, X_{i-1}]$.

Definition 3.32 (Triangular system). *Let $((n_i, P_i, p_i))_{i=1}^{\ell}$ such that for all i, n_i is a positive integer and $P_i \in \mathbb{Q}[X_1, \ldots, X_{i-1}][X_i]$ with $deg(P_i) = p_i > 0$. Let $(\alpha_1, \ldots, \alpha_\ell)$ be a sequence of reals. Then $((n_i, P_i, p_i))_{i=1}^{\ell}$ is a triangular system of level ℓ for $(\alpha_1, \ldots, \alpha_\ell)$ if:*

- *α_1 is the n_1^{th} root of P_1 whose degree is p_1;*
- *For $1 \leq i < \ell$, $P_{i+1}(\alpha_1, \ldots, \alpha_i)$ has degree p_i and α_{i+1} is the n_{i+1}^{th} root of polynomial $P_{i+1}(\alpha_1, \ldots, \alpha_i) \in \mathbb{Q}[\alpha_1, \ldots, \alpha_i][X_{i+1}]$.*

By convention, a triangular system of level 0 is the empty sequence. Observe that *a priori* we do not know how to decide whether $((n_i, P_i, p_i))_{i=1}^{\ell}$ is a triangular system for some sequence of reals. Given a triangular system $((n_i, P_i, p_i))_{i=1}^{\ell}$, a representation of an item of $\mathbb{Q}[\alpha_1, \ldots, \alpha_\ell]$ is nothing but some polynomial $P \in \mathbb{Q}[X_1, \ldots, X_l]$ denoting $P(\alpha_1, \ldots, \alpha_\ell)$.

Algorithm 7: Computing PmV in triangular systems

```
PmVPol($\ell, \mathcal{T}, P, p, Q, q$): an integer
Input: $\ell$, the current level
Input: $\mathcal{T} = \{(n_i, P_i, p_i)\}_{i=1}^{\ell}$ a triangular system for $(\alpha_1, \ldots, \alpha_\ell)$
Input: $P, Q$, polynomials $\mathbb{Q}[X_1, \ldots, X_\ell][X_{\ell+1}]$ of degree $p$ and $q$ with $q < p$
        such that $P(\alpha_1, \ldots, \alpha_\ell) \neq 0$ and $Q(\alpha_1, \ldots, \alpha_\ell) \neq 0$ when $q \geq 0$
Output: $PmV(sRes_p(P(\alpha_1, \ldots, \alpha_\ell), Q(\alpha_1, \ldots, \alpha_\ell)),$
        $\ldots, sRes_0(P(\alpha_1, \ldots, \alpha_\ell), Q(\alpha_1, \ldots, \alpha_\ell)))$
Data: $j$ an index, $s_p, \ldots, s_0$ a sequence of signs
if $q = -\infty$ then return 0 // consistently with Cauchy index definition
$sRes(P, Q) \leftarrow$ SubResultants($\mathbb{Q}[X_1, \ldots, X_\ell], P, p, Q, q$) // using Algorithm 2
for $j$ from 0 to $p$ do $s_j \leftarrow$ Sign($\ell, \mathcal{T}, sRes_j(P, Q)$)
return $PmV(s_p, \ldots, s_0)$ // by applying the definition
```

Example 3.33. The system $((2, X_1^2 - X_1 - 1, 2), (1, (2X_1 - 1)X_2^2 - 1, 2))$ is a triangular system for the reals $(\frac{1+\sqrt{5}}{2}, -\frac{1}{\sqrt[4]{5}})$. Indeed, polynomial $X_1^2 - X_1 - 1$ has two roots $\frac{1-\sqrt{5}}{2} < \frac{1+\sqrt{5}}{2}$. In addition, when $X_1 = \frac{1+\sqrt{5}}{2}$, polynomial $(2X_1 - 1)X_2^2 - 1$ becomes $P_1 = \sqrt{5}X_2^2 - 1$, with two roots $-\frac{1}{\sqrt[4]{5}} < \frac{1}{\sqrt[4]{5}}$.

Proposition 3.34. *Let $\ell \geq 0$ and $((n_i, P_i, p_i))_{i=1}^{\ell}$ such that for all i, n_i is a positive integer and $P_i \in \mathbb{Q}[X_1, \ldots, X_{i-1}][X_i]$ with $deg(P_i) = p_i > 0$. Then we can decide whether $((n_i, P_i, p_i))_{i=1}^{\ell}$ is a triangular system for some $\{\alpha_i\}_{i=1}^{\ell}$. Furthermore with this representation, the rings $\mathbb{Q}[\alpha_1, \ldots, \alpha_\ell]$ and $\mathbb{Z}[\alpha_1, \ldots, \alpha_\ell]$ are sign-effective.*

Proof. The proof is done by induction on ℓ. The base case $\ell = 0$ corresponds to the case where the ring is $\mathbb{Q}$ or $\mathbb{Z}$ and so there is nothing to prove.
For the inductive case, in order to check whether $((n_i, P_i, p_i))_{i=1}^{\ell+1}$ is a triangular system, we first check that $((n_i, P_i, p_i))_{i=1}^{\ell}$ is a triangular system. In the positive case $\mathbb{Q}[\alpha_1, \ldots, \alpha_\ell]$ is sign-effective so that we can check whether $P_{\ell+1}(\alpha_1, \ldots, \alpha_\ell)$ has degree $p_{\ell+1}$ and compute the number of roots of $P_{l+1}(\alpha_1, \ldots, \alpha_\ell)$ by using `PmvPol`$(l, \mathcal{T}, P_{l+1}, p_{l+1}, P'_{\ell+1}, p_{\ell+1} - 1)$ in $\mathbb{Q}[\alpha_1, \ldots, \alpha_\ell]$. We have rewritten the corresponding algorithm (see Algorithm 7) in order to exploit the representation provided by Algorithm 2.
Assume that $((n_i, P_i, p_i))_{i=1}^{\ell+1}$ is a triangular system. Again using induction hypothesis $\mathbb{Q}[\alpha_1, \ldots, \alpha_\ell]$ is sign-effective. So in addition to sign determination in $\mathbb{Q}[\alpha_1, \ldots, \alpha_\ell]$, we are also able to compute `Degree` and `RootCoding` in this ring. Thus Algorithm 8 (applied at level $l+1$) determines the sign of $P(\alpha_1, \ldots, \alpha_{\ell+1})$ by computing the degree of P in $\mathbb{Q}[\alpha_1, \ldots, \alpha_\ell]$ and then determining the P-encodings of roots of $P_{\ell+1}$ in $\mathbb{Q}[\alpha_1, \ldots, \alpha_\ell]$ and returning the sign of P corresponding to the $n_{\ell+1}^{th}$ root.

□

The sign determination is then obtained by a set of mutually recursive functions. In order to clarify their behavior we have represented their calls in Figure 4.

$$\begin{tikzcd}
 & \texttt{Sign} \arrow[d, "-1"] \arrow[ddrr, "-1"] & & \\
 & \texttt{RootCoding} \arrow[d] & & \\
\texttt{PmVPol} \arrow[uur] & \texttt{SignRealization} \arrow[l] \arrow[r] & \texttt{IntRem} \arrow[r] & \texttt{Degree} \arrow[uull]
\end{tikzcd}$$

Fig. 4. Links between function calls with level ℓ changing.

Algorithm 8: Determining the sign in a triangular system.

```
Sign(ℓ, 𝒯, P): a sign
Input: P, a polynomial in $\mathbb{Q}[X_1, \ldots, X_\ell] = \mathbb{Q}[X_1, \ldots, X_{\ell-1}][X_\ell]$
Input: $\ell$, the current level
Input: $\mathcal{T} = \{(n_i, P_i, p_i)\}_{i=1}^{\ell}$ a triangular system for $(\alpha_1, \ldots, \alpha_\ell)$
Output: the sign of $P(\alpha_1, \ldots, \alpha_\ell)$
Data: $\Sigma$ a list of sign vectors
if $\ell = 0$ then return Sign($\mathbb{Q}, P$) // $P$ is a rational
$p \leftarrow$ Degree($\ell - 1, \mathcal{T}_{\downarrow \ell-1}, P$)
// $\mathcal{T}_{\downarrow \ell-1}$ is the restriction of $\mathcal{T}$ at level $\ell - 1$
if $p = -\infty$ then return 0
$\Sigma$ = RootCoding($\ell - 1, \mathcal{T}_{\downarrow \ell-1}, P_\ell, p_\ell, P, p$)
Let $\mathbf{v}$ be the $n_\ell^{th}$ item of $\Sigma$
return $\mathbf{v}[0]$
```

3.3 Building a cylindrical algebraic decomposition

We have the following result [13]:

Theorem 3.35. *For every finite family of sets of polynomials $\mathcal{P} = \{\mathcal{P}_i\}_{i \leq n}$ such that $\mathcal{P}_i \subseteq \mathbb{Q}[X_1, \ldots, X_i]$, one can build a cylindrical algebraic decomposition of $\mathbb{R}^n$ adapted to $\mathcal{P}$ in 2EXPTIME.*

We devote the rest of the subsection to the proof of this theorem. The algorithm that builds the cylindrical algebraic decomposition of $\mathbb{R}^n$ proceeds in two steps: the *elimination step* and the *lifting step*. The elimination step ensures the existence of a cylindrical algebraic decomposition while enlarging the set of polynomials of polynomials $\mathcal{P}_i$. Once $\mathcal{P}$ has been completed, the lifting step provides an effective way to compute the cylindrical algebraic decomposition. Accordingly, one considers the coefficients of polynomials in $\mathbb{R}$ during the elimination step and restrict them to belong to $\mathbb{Q}$ during the lifting step.

Elimination step. The following lemma establishes that the roots of a polynomial are "continuous" w.r.t. the coefficients of the polynomial when the degree of the polynomial remains constant.

Lemma 3.36. *Let $P \in \mathbb{C}[X_1, \ldots, X_{k-1}][X_k]$, $S \subseteq \mathbb{C}^{k-1}$ such that $deg(P(x))$ is constant over $x \in S$. Let $a \in S$ such that $z_1, \ldots, z_m$ are the roots of $P(a)$ with multiplicities $\mu_1, \ldots, \mu_m$, respectively. Let $0 < r < \min_{i \neq j}(|z_i - z_j|/2)$. Then there exists an open neighborhood U of a such that for $x \in U$, $P(x)$ has exactly μ_i roots counted with multiplicities in the disc $D(z_i, r)$ for all $i \leq m$.*

Proof. Since the degree of P is constant we can divide the coefficients by the leading coefficient, obtaining a monic polynomial with same roots and multiplicities and coefficients being rational functions.
Assume that $P = X_k^\mu$. Consider $Q = X^\mu - \sum_{i<\mu} b_i X^i$ with $\delta = \max_{i<\mu} |b_i| < \frac{\min(1,r^\mu)}{\mu}$. Since $\delta < \frac{1}{\mu}$, any root of Q has a module less than one. Let z be such a root. Then $z^\mu = \sum_{i<\mu} b_i z^i$. So $|z^\mu| \leq \mu\delta < r^\mu$ which implies $|z| < |r|$.
Let us consider the mapping from pairs (Q, R) of monic polynomials of degree respectively q and r to their product $\varphi(Q, R) = QR$ of degree $q + r$ (viewed as mapping of their coefficients). This mapping is differentiable. It is routine to check that the Jacobian matrix of this mapping is equal or opposite to the subresultant $Sres_0(Q, R)$ and so it locally admits a differentiable inverse if Q and R are coprime. Therefore, factoring $P = QR$ such that Q and R are coprime, there exists some neighborhoods $\mathcal{V}_Q$, $\mathcal{V}_R$ respectively of Q and R, such that $\mathcal{V} = \varphi(\mathcal{V}_Q \times \mathcal{V}_R)$ is a neighborhood of P.
By iteration, the polynomial $P_0 = (X_k - z_1)^{\mu_1} \cdots (X_k - z_m)^{\mu_m}$ admits an open neighborhood $\mathcal{V}$ of its coefficients such that every monic polynomial $P_1 \in \mathcal{V}$ admits a decomposition $P_1 = Q_1 \ldots Q_m$ with every Q_i of degree μ_i and whose roots belong to the disc $D(z_i, r)$. Since the discs have no intersection, every disc contains exactly μ_i roots counted with multiplicities.
Since the coefficients of P are rational functions of $X_1, \ldots X_{k-1}$ and so continuous, there is a neighborhood U of a that fulfills the conclusion of the lemma. $\square$

The next proposition establishes that the real roots of a set of polynomials are "continuous" w.r.t. the coefficients of the polynomials when the degrees of some appropriate polynomials (including the original ones) remain constant.

Proposition 3.37. *Let $P_1, \ldots, P_s \in \mathbb{R}[X_1, \ldots, X_{k-1}][X_k]$, $S \subseteq \mathbb{R}^{k-1}$ connected. Assume that over $x \in S$, for all $1 \leq i, j \leq s$, $P_i(x)$ is not identically 0, $deg(P_i(x))$, $deg(gcd(P_i(x), P_j(x))$, $deg(gcd(P_i(x), P_i'(x))$ are both constant.*
Then there exist ℓ (with ℓ possibly null) continuous functions $f_1 < \cdots < f_\ell$ from S to $\mathbb{R}$ such that for every $x \in S$, the set of real roots of $\prod_{j \leq s} P_j(x)$ is exactly $\{f_1(x), \ldots, f_\ell(x)\}$.
Moreover for all $i \leq \ell, j \leq s$, the multiplicity of the (possible) root $f_i(x)$ of $P_j(x)$ is constant over $x \in S$.

Proof. Let $a \in S$ and $z_1(a), \ldots, z_m(a)$ be the roots in $\mathbb{C}$ of $\prod_{j \leq s} P_i(a)$ with μ_i^j being the multiplicity of $z_i(a)$ for $P_j(a)$. The degree of $R_{jk}(a) = gcd(P_j(a), P_k(a))$ is $\sum_{i \leq m} \min(\mu_i^j, \mu_i^k)$ and $\min(\mu_i^j, \mu_i^k)$ is the (possibly null) multiplicity of $z_i(a)$ for $R_{jk}(a)$.
Pick $r > 0$ such that the discs $D(z_i(a), r)$ are disjoint. Observe that since $deg(gcd(P_j(x), P_j'(x))$ is constant over $x \in S$ the number of distinct roots of $P_j(x)$ is constant over $x \in S$. Let i, j such that $\mu_i^j > 0$, applying Lemma 3.36 and the previous observation, there is a neighborhood U of a such that for all $x \in U$, $D(z_i, r)$ contains exactly a root, denoted $z_i^j(x)$, of $P_j(x)$ with multiplicity μ_i^j. Assume there exists $k \neq j$ with $\mu_i^k > 0$, since $deg(R_{jk}(x))$ is constant over $x \in S$, $z_i^j(x) = z_i^k(x)$ for all $x \in U$. Otherwise for such an x where the equality does not hold $deg(R_{jk}(x)) < deg(R_{jk}(a))$. So we can omit the superscript j in $z_i^j(x)$ (defined when $\mu_i^j > 0$).
If $z_i(a)$ is real then $z_i(x)$ is real otherwise its conjugate would be another root in $D(\overline{z_i(a)}, r)$. If $z_i(a)$ is complex, its conjugate being also a root, $D(z_i(a), r)$ and $D(\overline{z_i(a)}, r)$ are disjoint and so $z_i(x)$ is not real. Hence the number of real roots of (x) is constant over $x \in U$. As the number of real roots is locally constant and S is connected then the number of real roots of $\prod_{j \leq s} P_j(x)$ is constant over $x \in S$, say ℓ.
Let $f_i(x)$, for $i \leq l$ be the function that associates with x the i^{th} real root of $\prod_{j \leq s} P_j(x)$ in increasing order. Since r could be chosen arbitrarily small, f_i is continuous. As the multiplicity of $f_i(x)$ w.r.t. any $P_j(x)$ and $Q(x)$ is locally constant, it is constant over $x \in S$. □

The next definition is a basic construction that will be the atomic step of the elimination stage.

Definition 3.38. *Let $P = \sum_{i \leq p} a_i X_k^i \in \mathbb{R}[X_1, \ldots, X_{k-1}][X_k]$. Then $lcof(P) = a_p$ and $Tru(P) = \{\sum_{i \leq r} a_i X_k^i \mid \forall i > r \ a_i \notin \mathbb{R}^* \wedge a_r \neq 0\}$.*
Let $\mathcal{P}$ be a finite subset of $\mathbb{R}[X_1, \ldots, X_{k-1}][X_k]$. Then $Elim_{X_k}(\mathcal{P})$ is the set of polynomials of $\mathbb{R}[X_1, \ldots, X_{k-1}]$ defined as follows. For all $P, Q \in \mathcal{P}, R \in Tru(P)$, $T \in Tru(Q)$ with $deg(T) \leq deg(R)$:

- *If $lcof(R)$ does not belong to $\mathbb{R}$ then $lcof(R) \in Elim_{X_k}(\mathcal{P})$;*
- *If $deg(R) \geq 2$ then for all $sRes_j(R, R')$ that are defined and do not belong to $\mathbb{R}$, $sRes_j(R, R') \in Elim_{X_k}(\mathcal{P})$;*
- *for all $sRes_j(R, T)$ that are defined and do not belong to $\mathbb{R}$, $sRes_j(R, T) \in Elim_{X_k}(\mathcal{P})$.*

The next lemma establishes the interest of the $Elim_{X_k}$ construction.

Lemma 3.39. *Let $\mathcal{P}$ be a finite set of $\mathbb{R}[X_1, \ldots, X_{k-1}][X_k]$, $S \subseteq \mathbb{R}^{k-1}$ a connected set. Assume that S is $Elim_{X_k}(\mathcal{P})$-invariant.*
Then there exist ℓ (with ℓ posibly null) continuous functions $f_1 < \cdots < f_\ell$ from S to $\mathbb{R}$ such that for every $x \in S$, the set of real roots of $\prod_{P \in \mathcal{P}^*} P(x)$ is exactly $\{f_1(x), \ldots, f_\ell(x)\}$ where $\mathcal{P}^*$ is the subset of $\mathcal{P}$ consisting of polynomials not identically null over S.
Moreover for all $i \leq l$ and for all $P \in \mathcal{P}^*$, the multiplicity of the root $f_i(x)$ of $P(x)$ is constant over $x \in S$.

Proof. Let $P \in \mathcal{P}$. Since the leading coefficients of $Tru(P)$ belong to $Elim_{X_k}(\mathcal{P})$, the degree of $P(x)$ is constant over $x \in S$.

Let $R \in Tru(P)$ be the *appropriate* polynomial for P (i.e. whose degree is the degree of $P(x)$ for $x \in S$). Then, by $deg(gcd(R, R'))$ is determined by the signs of polynomials of the sequence $Sres(R, R')$ due to Proposition 3.7. Since all these polynomials belong to $Elim_{X_k}(\mathcal{P})$, the number of distinct complex roots of $deg(gcd(P(x), P'(x))$ is constant over $x \in S$.

Let $T \in Tru(Q)$ be the appropriate polynomial of Q for $Q \in \mathcal{P}$. Then, by Proposition 3.7, $deg(gcd(R, T))$ is determined by the signs of polynomials of the sequence $Sres(R, T)$. Since all these polynomials belong to $Elim_{X_k}(\mathcal{P})$, the degree of $gcd(P(x), Q(x))$ is constant over $x \in S$.

The conclusion follows using Proposition 3.37.

□

We are now in position define the elimination step and to prove its correctness.

Theorem 3.40. *Let $\mathcal{Q} = \{\mathcal{Q}_i\}_{i \leq n}$ be a family of finite set of polynomials such that $\mathcal{Q}_i \subseteq \mathbb{R}[X_1, \ldots, X_i]$. Define $\mathcal{P}_n = \mathcal{Q}_n$ and inductively $\mathcal{P}_{i-1} = \mathcal{Q}_{i-1} \cup Elim_{X_i}(\mathcal{Q}_i)$ for $i > 1$. Then there exists a cylindrical algebraic decomposition adapted to $\mathcal{P}$ (and thus to $\mathcal{Q}$).*

Proof. Let us prove the existence of a cylindrical algebraic decomposition of $\mathbb{R}^i$ adapted to $\mathcal{P}_i$ by induction.
The children of $\mathbb{R}^0$ form the partition defined by

$$(-\infty, r_1), r_1, (r_1, r_2), \ldots, (r_{m-1}, r_m), r_m, (r_m, \infty)$$

where $\{r_1, \ldots, r_m\}$ is the set of roots of all $P \in \mathcal{P}_1$ (or $\mathbb{R}$ if there is no root). By construction, the cells of $\mathcal{S}_1$ are $\mathcal{P}_1$-invariant and open intervals or points.
Assume that we have built our tree up to level $i < n$. Pick any cell C of level i. C is $Elim_{X_{i+1}}(\mathcal{P}_{i+1})$-invariant since $Elim_{X_{i+1}}(\mathcal{P}_{i+1}) \subseteq \mathcal{P}_i$. Applying Lemma 3.39 yields the children of C.

□

Complexity of elimination step. Let $s = |\mathcal{Q}|$, d be the maximal total degree of polynomials of $\mathcal{Q}$, and v the maximal constant appearing in a coefficient of $\mathcal{Q}$. A straightforward recurrence shows that

– the maximal number of bits of a coefficient of any $\mathcal{P}_i$ is $O(d^n \cdot 3^{\log(n)\cdot \frac{n(n-1)}{2}} \cdot \log(v))$,
– the maximal total degree of polynomials of all $\mathcal{P}_i$ is in $O(d^{3^n})$, and
– the total number of polynomials is in $O((sd)^{3^n})$.

Example 3.41. Let us build the family $\mathcal{P}_1, \mathcal{P}_2$ of polynomials associated with the automaton of Fig. 1. We set $I_1 = X_1$, $I_2 = X_2$, $A = X_1^2 - X_1 - 1$, $B = (2X_1 - 1)X_2^2 - 1$ and $C = X_2 + (X_1^2 - 5)$. We start with $\mathcal{P}_2 = \{I_2, B, C\}$, $\mathcal{P}_1 = \{I_1, A\}$ and add to $\mathcal{P}_1$ polynomials computed by $Elim_{X_2}(\mathcal{P}_2)$.

We first add $lcof(B) = 2X_1 - 1$ to $\mathcal{P}_1$. Note that we do not add $lcof(C)$ since it is in $\mathbb{Q}$.

Let us now compute all subresultants of (potentially truncated) polynomials of $\mathcal{P}_2$:

- $sRes_0(I_2, C) = \begin{vmatrix} 1 & 0 \\ 1 & X_1^2 - 5 \end{vmatrix} = X_1^2 - 5$ is added to $\mathcal{P}_1$.
- We then add to $\mathcal{P}_1$ the polynomial

$$\begin{aligned} sRes_0(B, C) &= \begin{vmatrix} 2X_1 - 1 & 0 & -1 \\ 0 & 1 & X_1^2 - 5 \\ 1 & X_1^2 - 5 & 0 \end{vmatrix} \\ &= -(2X_1 - 1)(X_1^2 - 5)^2 + 1 \\ &= -2X_1^5 + X_1^4 + 20X_1^3 - 10X_1^2 - 50X_1 + 26 \end{aligned}$$

- Remark that $sRes_0(B, I_2) = 1 \in \mathbb{Q}$, hence it is not added to $\mathcal{P}_1$. It is also the case for $sRes_1(B, I_2)$ and $sRes_1(B, C)$.

We then need to compute the subresultants of each polynomial of degree ≥ 2 with its derivative. In our case, that means computing $sRes_0(B, B')$ and $sRes_1(B, B')$. We have $B' = 2(2X_1-1)X_2$. We obtain $sRes_1(B, B') = 2(2X_1-1)$ that should be added to $\mathcal{P}_1$. However, since $sRes_1(B, B') = 2lcof(B)$, their sign will coincide. For simplicity we will not keep it in $\mathcal{P}_1$, although the automatic procedure does; nonetheless, this would not affect the elimination at lower levels. Finally, we have

$$sRes_0(B, B') = \begin{vmatrix} 2X_1 - 1 & 0 & -1 \\ 0 & 2(2X_1 - 1) & 0 \\ 2(2X_1 - 1) & 0 & 0 \end{vmatrix} = 4(2X_1 - 1)^2$$

which is added to $\mathcal{P}_1$. This concludes the elimination phase.

The final sets $\mathcal{P}_1$ and $\mathcal{P}_2$ are given in Table 1 (page 40).

Algorithm 9: Lifting the cylindrical algebraic decomposition at a point of level ℓ

```
Input: $\mathcal{P} = \{\mathcal{P}_\ell\}_{\ell \leq k}$ a family of subsets of polynomials obtained by
        decomposition
Output: $\mathcal{A}$ a tree whose nodes at level $\ell$ are sample points of the
        decomposition equipped with their sign evaluation for $\mathcal{P}_\ell$
Lifting($\ell, \mathcal{T}$): an integer
Input: $\ell$, the current level; $\mathcal{T} = \{(n_i, P_i, p_i)\}_{i=1}^{\ell}$ a triangular system for
        $(\alpha_1, \ldots, \alpha_\ell)$ corresponding to a node of $\mathcal{A}$.
Data: $L$ a list of triangular systems equipped with sign vectors, $E$ a triangular
        system with a sign vector
$L \leftarrow$ LinePartition($\ell, \mathcal{T}$)
if $L = \emptyset$ then
    $\mathcal{T}' \leftarrow \mathcal{T} \cup \{(1, X_{\ell+1}, 1)\}$; $\mathcal{T}' \cdot Eval \leftarrow \{(P, \texttt{Sign}(\ell, \mathcal{T}, \texttt{Lcof}(P)) \mid P \in \mathcal{P}_{\ell+1}\}$
    $\mathcal{A} \leftarrow \mathcal{A} \cup (\mathcal{T} \to \mathcal{T}')$; if $\ell + 1 < k$ then Lifting($\ell + 1, \mathcal{T}'$)
else
    $L \leftarrow$ Completing($\ell, \mathcal{T}, L$)
    for $E \in L$ do
        Pick some $(r, v, P) \in E$ such that $r$ is defined
        $\mathcal{T}' \leftarrow \mathcal{T} \cup \{(r, P, \texttt{Degree}(\ell, \mathcal{T}, P))\}$;
        $\mathcal{T}' \cdot Eval \leftarrow \{(Q, v[0]) \mid Q \in \mathcal{P}_{\ell+1} \wedge \exists (m, v, Q) \in E\}$
        $\mathcal{A} \leftarrow \mathcal{A} \cup (\mathcal{T} \to \mathcal{T}')$; if $\ell + 1 < k$ then Lifting($\ell + 1, \mathcal{T}'$)
    end
end
```

Lifting step. We build the cylindrical algebraic decomposition as follows: every cell C of level ℓ is represented by a *sample point*, represented by a triangular system. In addition, the representation of C includes the evaluation of the sign of all $P \in \mathcal{P}_\ell$. Observe that evaluation of a $P \in \mathcal{P}_j$ with $j < \ell$ is found in its ancestor cell of level j. The construction is performed by Algorithm 9. An atomic step of the lifting phase corresponds to build, given a sample point $\mathbb{R}^\ell$, the ordered list of all sample points of $\mathbb{R}^{\ell+1}$ representing the cells of the cylinder above S. It corresponds to a call to `Lifting` (without the recursive calls). The whole construction is done by the call `Lifting`$(0, \emptyset)$. `Lifting` first calls `LinePartition` in order to get an ordered list of the roots of all $P \in \mathcal{P}_{\ell+1}$. Every real α of this list is represented by a set of triplets (r, v, P) where P is a polynomial whose coefficients are algebraic numbers over $\mathcal{T}$ (and thus represented by polynomials in $Q[X_1, \ldots, X_\ell]$), v is the P-encoding of α. r may be undefined but when defined it means that α is the r^{th} root of P. For at least one triplet of the set r is defined allowing to extend the triangular system $\mathcal{T}$ by α. Since one wants to represent the interval between these roots by sample points, the list is completed by a call to `Completing`. After this call either the list is empty (corresponding to the case of a single child $C \times \mathbb{R}$) and this child is represented by $\alpha_{\ell+1} = 0$, first root of $X_{\ell+1}$. The representation of this cell is now enlarged by the evaluation of all $P \in \mathcal{P}_{\ell+1}$ at this sample point. Otherwise for every item of the list one picks some arbitrary (r, v, P) with r defined and proceeds as previously to produce all the children of C.

Algorithm 10 produces the list of roots of all $P(\alpha_1, \ldots, \alpha_\ell)$ for $P \in \mathcal{P}_{\ell+1}$. For any such P, it first normalizes it by determining its higher non null coefficient. Thus $R \in Tru(P)$. $SL[P]$ will contain the singletons $\{(r, v, P)\}$ for every root of $P(\alpha_1, \ldots, \alpha_\ell)$. Then the algorithm enlarges these singletons with triplets $\{(r', v', Q)\}$ for all Q that preceed P in $\mathcal{P}_{\ell+1}$. All these triplets are obtained using the lists provided by appropriate calls to `RootCoding`. Conversely the sets of the list $SL[Q]$ are enlarged with the triplets related to P. Once all roots have been produced in SL, it remains to order them and (possibly) merge them. This can be easily done with the help of their Thom-encoding and it is performed by a call to `OrderedMerge`.

Algorithm 11 completes the list of roots by sample points representing the intervals between the roots. This is done as follows. Given a root α of P and a root β of Q, such that α and β are consecutive items of the list, there exists a root f of $(PQ)'$ such that $f \in]\alpha, \beta[$. Thus the sample point will be an arbitrary root of $(PQ)'$ strictly between α and β. If α is the smallest (resp. largest) root in the list for of some P then the first (resp. last) root of $P[X_{\ell+1} + 1]$ (resp. $P[X_{\ell+1} - 1]$) is $\alpha - 1 \in] - \infty, \alpha[$ (resp. $\alpha + 1 \in]\alpha, +\infty[$). In this algorithm E represents the current item, say β of the list of roots, P some polynomial whose β is a root and v is its P-encoding. Let α be the previous item of the list (when it exists). $oldP$ is some polynomial whose α is a root and $oldv$ is its $oldP$-encoding. Thus in order to find a root of $(P \cdot oldP)'$ between α and β, one computes the P and $oldP$ encoding of the roots of $(P \cdot oldP)'$.

Algorithm 10: Partitioning the real line at a point of level ℓ.

```
Input: $\mathcal{P} = \{\mathcal{P}_\ell\}_{\ell \le k}$ a family of subsets of polynomials
LinePartition($\ell, \mathcal{T}$): a list
Input: $\ell$, the current level
Input: $\mathcal{T} = \{(n_i, P_i, p_i)\}_{i=1}^{\ell}$ a triangular system for $(\alpha_1, \ldots, \alpha_\ell)$ corresponding
        to a node of $\mathcal{A}$ whose children have to be computed.
Output: $L$ a list of sample points of the decomposition equipped with their
        sign evaluation for $\mathcal{P}_{\ell+1}$ related to $\mathcal{T}$.
for $P \in \mathcal{P}_{\ell+1}$ do
    $(R, r) \leftarrow$ Normalize($\ell, \mathcal{T}, P$)
    if $r \le 0$ then $SL[P] \leftarrow \emptyset$
    else
        $SLL \leftarrow$ RootCoding($\ell, \mathcal{T}, R, r, R, r$)
        // Singleton transforms a list of items into a list of
           singletons which contain these items. Furthermore it adds
           the number of the root of $R$ for subsequent use.
        $SL[P] \leftarrow$ Singleton($SLL$)
        for $Q \in \mathcal{P}_{\ell+1}$ such that $Q \prec P$ do
            $(S, s) \leftarrow$ Normalize($\ell, \mathcal{T}, Q$)
            $SLL \leftarrow$ RootCoding($\ell, \mathcal{T}, R, r, S, s$); EnlargeWith($SL[P], SLL, Q$)
        end
    end
    for $Q \in \mathcal{P}_{\ell+1}$ such that $Q \prec P$ do
        if $SL[Q] \ne \emptyset$ then
            $(S, s) \leftarrow$ Normalize($\ell, \mathcal{T}, Q$)
            $SLL \leftarrow$ RootCoding($\ell, \mathcal{T}, S, s, R, r$); EnlargeWith($SL[Q], SLL, P$)
        end
    end
end
$L \leftarrow$ OrderedMerge($SL$)
return $L$
```

Algorithm 11: Completing the line partition with samples of intervals.

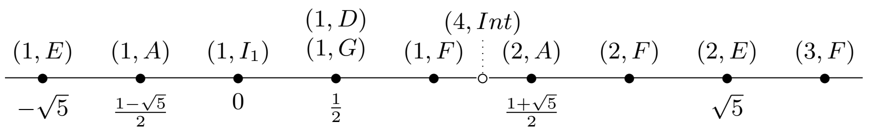
```
Input: $\mathcal{P} = \{\mathcal{P}_\ell\}_{l \leq k}$ a family of subsets of polynomials obtained by
        decomposition
Completing($\ell, \mathcal{T}, L$): a list
Input: $\ell$, the current level
Input: $\mathcal{T} = \{(n_i, P_i, p_i)\}_{i=1}^{\ell}$ a triangular system for $(\alpha_1, \ldots, \alpha_\ell)$ corresponding
        to a node of $\mathcal{A}$ whose children have to be computed.
Input: $L$ a list of sample points of the decomposition represented by a
        triangular system equipped with their sign evaluation for $\mathcal{P}_{\ell+1}$ related
        to $\mathcal{T}$.
Output: the input list $L$ enriched with of sample points for the intervals before,
         between and beyond the original sample points.
for $E \in L$ do
    Pick some $(r, v, P) \in E$ such that $r$ is defined
    if $E =$ First($L$) then
        $(R, r) \leftarrow$ Normalize($\ell, \mathcal{T}, P(X_{\ell+1} + 1)$);
        $SLL \leftarrow$ RootCoding($\ell, \mathcal{T}, R, r, R, r$)
        $shortL \leftarrow$ Singleton($SLL$)
        for $Q \in \mathcal{P}_{\ell+1}$ do
            $(S, s) \leftarrow$ Normalize($\ell, \mathcal{T}, Q$)
            $SLL \leftarrow$ RootCoding($\ell, \mathcal{T}, R, r, S, s$); EnlargeWith($shortL, SLL, Q$)
        end
        Insert First($shortL$) before $E$ in $L$
    else
        $(R, r) \leftarrow$ Normalize($\ell, \mathcal{T}, (P \cdot oldP)'$; $SLL \leftarrow$ RootCoding($\ell, \mathcal{T}, R, r, R, r$)
        $shortL \leftarrow$ Singleton($SLL$)
        for $Q \in \mathcal{P}_{\ell+1}$ do
            $(S, s) \leftarrow$ Normalize($\ell, \mathcal{T}, Q$)
            $SLL \leftarrow$ RootCoding($\ell, \mathcal{T}, R, r, S, s$); EnlargeWith($shortL, SLL, Q$)
        end
        Find $F$ in $shortL$ such that $\exists(x, vP, P), (y, voldP, oldP) \in F$
        with $vP < v$ and $voldP > oldv$; Insert $F$ before $E$ in $L$
    end
    $oldv \leftarrow v$; $oldP \leftarrow P$
end
Let $E$ be Last($L$)
Pick some $(r, v, P) \in E$ such that $r$ is defined
$(R, r) \leftarrow$ Normalize($\ell, \mathcal{T}, P(X_{\ell+1} - 1)$)
$SLL \leftarrow$ RootCoding($\ell, \mathcal{T}, R, r, R, r$); $shortL \leftarrow$ Singleton($SLL$)
for $Q \in \mathcal{P}_{\ell+1}$ do
    $(S, s) \leftarrow$ Normalize($\ell, \mathcal{T}, Q$)
    $SLL \leftarrow$ RootCoding($\ell, \mathcal{T}, R, r, S, s$); EnlargeWith($shortL, SLL, Q$)
end
Insert Last($shortL$) after $E$ in $L$; return $L$
```

Example 3.42. We first (by Algorithm 10) compute the line partition of $\mathbb{R}$ at level 1 for $\mathcal{P}_1 = \{I_1, A, D, E, F, G\}$ (see Table 1) obtained previously. This is done by comparing the P-encodings of roots of Q for all pairs $(P, Q) \in \mathcal{P}_1^2$. The result is (partially) depicted in Fig. 5. Each bullet represents the (relative) position of a root, given by a triangular system (where the degree of the polynomial is not represented for clarity). In the table, the line labeled by P gives the P-encodings of the roots.

$$\begin{aligned}
I_1 &= X_1\\
I_2 &= X_2\\
A &= X_1^2 - X_1 - 1\\
B &= (2X_1 - 1)X_2^2 - 1\\
C &= X_2 + X_1^2 - 5\\
D &= 2X_1 - 1 \ (= lcof(B))\\
E &= X_1^2 - 5 \ (= sRes_0(I_2, C))\\
F &= -2X_1^5 + X_1^4 + 20X_1^3 - 10X_1^2 - 50X_1 + 26 \ (= sRes_0(B, C))\\
G &= 4(2X_1 - 1)^2 \ (= sRes_0(B, B'))\\
Int &= -14X_1^6 + 18X_1^5 + 105X_1^4 - 124X_1^3 - 180X_1^2 + 172X_1 + 24 \ (= (FA)')
\end{aligned}$$

$$\mathcal{P}_1 = \{I_1, A, D, E, F, G\} \qquad\qquad \mathcal{P}_2 = \{I_2, B, C\}$$

Table 1. Polynomials used in the cylindrical decomposition.



I_1	$(-1,1)$	$(-1,1)$	$(0,1)$	$(1,1)$	$(1,1)$	$(1,1)$	$(1,1)$	$(1,1)$	$(1,1)$
A	$(1,-1,1)$	$(0,-1,1)$	$(-1,-1,1)$	$(-1,0,1)$	$(-1,1,1)$	$(0,1,1)$	$(1,1,1)$	$(1,1,1)$	$(1,1,1)$
D			$(-1,1)$						
E	$\vdots$		$(-1,-1,1)$				$\ddots$		
F			$(1,1,-1,1,1,-1)$						
G			$(1,-1,1)$						

Fig. 5. Partition of $\mathbb{R}$ according to $\mathcal{P}_1$ and Thom encodings. The scale is not accurate.

Example 3.43. We can now complete the line built above by computing sample points corresponding to intervals between consecutive roots (Algorithm 11). For instance to compute a sample point at the left of $(1, E) = -\sqrt{5}$, one can choose $-1-\sqrt{5}$ which is the first root of $H = (X+1)^2 - 5$ (*i.e.* E where X is replaced by $X + 1$). In order to compute a value between $(1, F)$ and $(2, A)$, we consider the polynomial $Int = (FA)' = -14X_1^6 + 18X_1^5 + 105X_1^4 - 124X_1^3 - 180X_1^2 + 172X_1 + 24$. Computing the F-encodings of roots of Int gives the number k of roots of Int smaller than or equal to $(1, F)$. Taking the $k + 1$th root of Int yields a root greater than $(1, F)$. The value $(k + 1, Int)$ is smaller than $(2, A)$ (since one such root exists). Here, one can show that the appropriate root is the 4th. Hence the sample point $(4, Int)$ written α_1 is added to the line in order to represent interval $](1, F), (2, A)[$, as depicted by the empty bullet on Fig. 5. In addition, for all polynomials P of $\mathcal{P}_1$, the P-encoding of $(4, Int)$ is computed: the first component yields the sign of P in the interval. Namely:

$$I_1(\alpha_1) > 0 \qquad A(\alpha_1) < 0 \qquad D(\alpha_1) > 0$$

$$E(\alpha_1) < 0 \qquad F(\alpha_1) < 0 \qquad G(\alpha_1) > 0$$

Remark that this interval corresponds to the one where transition a of Fig. 1 is fired in the trajectory of Fig. 2.

Sample points (and their encodings) for all intervals should be computed and added to the line. This is omitted for readability.

Example 3.44. We illustrate the lifting (Algorithm 9) to $\mathbb{R}^2$ for the interval represented by the sample point $(4, Int)$ built above. In this case, one must partition the real line with roots of polynomials of $\mathcal{P}_2 = \{I_2, B, C\}$ when $X_1 = \alpha_1$. Note that I_1 and A are constants.

In the computation of the $\mathcal{P}_2$-encodings, the $\mathcal{P}_1$-encodings of α_1 are used, in particular the encodings of polynomials constructed in the elimination phase. For example, since $D(\alpha_1) > 0$, the leading coefficient of B is positive, hence B has two roots. And since $E(\alpha_1) < 0$, the root of $C(\alpha_1)$ is positive (greater than the root of I_2). Finding that all the roots of $B(\alpha_1)$ are smaller than γ the root of $C(\alpha_1)$ involves not only the sign of $F(\alpha_1)$ (which only shows that γ is not between the roots of $B(\alpha_1)$) but additional components of the encoding, namely in this case the sign of the second derivative of F. This is partially represented in Fig. 6 (again, the degrees of the polynomials are omitted). Note that this lifting corresponds to the trajectory depicted in Fig. 2, page 5.

4 Verification algorithms for PolITA

We now use the cylindrical decomposition to build a finite abstraction of the transition system associated with a PolITA. The model checking problem (hence also the reachability problem) can be solved with this abstraction. An on-the-fly construction is then given to produce a more efficient practical algorithm. Formally, we prove the following:

Theorem 4.1. *The model checking problem of TCTL_{int} over* PolITA *is decidable in time $(|\mathcal{A}| \cdot |\psi| \cdot d)^{2^{O(n)}}$ where n is the number of clocks in $\mathcal{A}$ and d the maximal degree of polynomials appearing in $\mathcal{A}$ and ψ.*

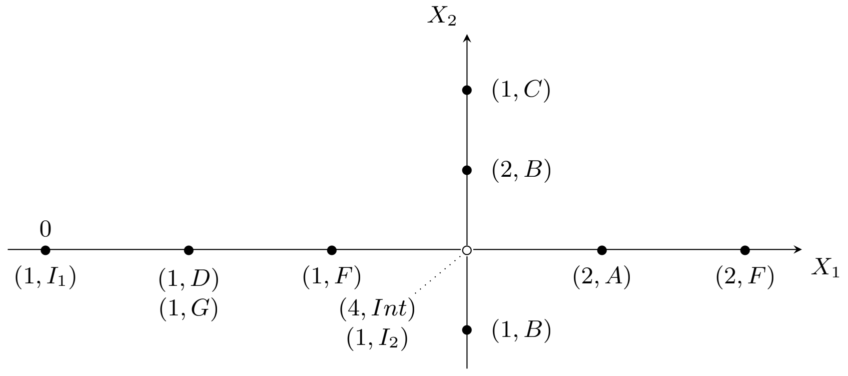


Fig. 6. Line partitioning for X_2 above $\alpha_1 = (4, Int)$.

4.1 Abstraction construction

Let $\mathcal{A} = \langle \Sigma, Q, q_0, F, X, \lambda, \Delta \rangle$ be a PoLITA with $X = \{x_1, \ldots, x_n\}$. We define $Poly(\mathcal{A})$ the set of all polynomials appearing in guards and updates of $\mathcal{A}$ (including all clocks) as follows:

$$Poly(\mathcal{A}) = X \cup \bigcup_{(q,g,a,u,q') \in \Delta} \left(\left\{ \bigcup_i \{P_i\} \;\middle|\; \varphi = \bigwedge_i P_i \bowtie_i 0 \right\} \right.$$
$$\left. \cup \left\{ \bigcup_{i=1}^{n} \{x_i - P_i\} \;\middle|\; u = \bigwedge_{i=1}^{n} x_i := P_i \right\} \right)$$

Given a $\mathsf{TCTL}_{\mathrm{int}}$ formula ψ, we define $Poly(\psi)$ the set of all polynomials appearing in ψ, *i.e.* in subformulas of the form $P \bowtie 0$. Note that in the case of the reachability problem, $Poly(\psi) = \emptyset$.

Let $\mathcal{D}_{\mathcal{A},\psi}$ be the cylindrical algebraic decomposition adapted to $Poly(\mathcal{A}) \cup Poly(\psi)$ and X. Since $\mathcal{D}_{\mathcal{A},\psi}$ is adapted to X, the cells can be arranged in levels $\mathcal{D}^1_{\mathcal{A},\psi}, \ldots, \mathcal{D}^n_{\mathcal{A},\psi}$, such that for $1 \leq i \leq n$, $\bigcup_{k=1}^{i} \mathcal{D}^k_{\mathcal{A},\psi}$ is a CAD of $\mathbb{R}^{\{x_1,\ldots,x_i\}}$. As a result, the projection of a cell of level i over the axis $x_i = 0$ yields a cell of level $i - 1$.

We define $\mathcal{R}_{\mathcal{A},\psi}$ the finite transition system with states in $Q \times \mathcal{D}_{\mathcal{A},\psi}$, specifically, they can also be arranged by layer, with respect to the level of the state: $\bigcup_{i=1}^{n} \lambda^{-1}(i) \times \mathcal{D}^i_{\mathcal{A},\psi}$. Indeed, given a configuration (q, v) with $\lambda(q) = k$, the semantics of PoLITA require that for $k < i \leq n$, $v(x_i) = 0$, hence v belongs to a cell of $\mathcal{D}^k_{\mathcal{A},\psi}$. We now define the transitions of $\mathcal{R}_{\mathcal{A},\psi}$ as follows.

Time successors. Let $succ \notin \Sigma$ be a letter representing time elapsing. Let (q, C) be a state of $\mathcal{R}_{\mathcal{A},\psi}$, with $\lambda(q) = k$, and let $\underline{C} \in \mathcal{D}^{k-1}_{\mathcal{A},\psi}$ be the projection of C onto $\mathbb{R}^{k-1}$ and $-\infty = f_0 < \cdots < f_{r+1} = +\infty$ be the functions dividing $\underline{C}$ as in Definition 3.2. The *succ* transitions are defined as follows:

– if $C = \{(x, f_i(x)) \mid x \in \underline{C}\}$ for some $i \in \{1, \ldots, r\}$, then there is a transition $(q, C) \xrightarrow{succ} (q, C')$ where $C' = \{(x, y) \mid x \in \underline{C}, f_i(x) < y < f_{i+1}(x)\}$;
– if $C = \{(x, y) \mid x \in \underline{C}, f_{i-1}(x) < y < f_i(x)\}$ for some $i \in \{1, \ldots, r\}$, then there is a transition $(q, C) \xrightarrow{succ} (q, C')$ where $C' = \{(x, f_i(x)) \mid x \in \underline{C}\}$;
– otherwise, $C = \{(x, y) \mid x \in \underline{C}, f_r(x) < y < f_{r+1}(x)\}$, and there is a self-loop labeled by *succ*: $(q, C) \xrightarrow{succ} (q, C)$.

In all the above cases, C' is called the *time successor* of C (in the last case, C is its own time successor).

Proposition 4.2 (Correctness w.r.t. time elapsing). *Let v be a valuation of a cell C of level k.*

– *There exists $d > 0$ such that the elapsing of d time units for x_k yields a valuation $v +_k d \in C'$, the time successor of C.*
– *For any $0 < d' < d$, the elapsing of d' time units for x_k yields a valuation $v +_k d$ that is either in C or in C'.*

Proof. We again distinguish the possible cases for C:

– If $C = \{(x, f_i(x)) \mid x \in \underline{C}\}$ for some $i \in \{1, \ldots, r\}$, then the time successor $C' = \{(x, y) \mid x \in \underline{C}, f_i(x) < y < f_{i+1}(x)\}$. Then $v = (x, f_i(x))$. By elapsing $\frac{f_{i+1}(x) - f_i(x)}{2}$ time units in level k, one clearly obtains a valuation of C'. Moreover, for every inferior delay d', $v +_k d'$ is also in C'.
– If $C = \{(x, y) \mid x \in \underline{C}, f_{i-1}(x) < y < f_i(x)\}$ for some $i \in \{1, \ldots, r\}$, then $C' = \{(x, f_i(x)) \mid x \in \underline{C}\}$. Then $v = (x, y)$ with $f_{i-1}(x) < y < f_i(x)$. By elapsing $f_i(x) - y$ time units in level k, one clearly obtains a valuation of C'. Moreover, for every inferior delay d', $v +_k d'$ remains in C.
– Otherwise, $C = \{(x, y) \mid x \in \underline{C}, f_r(x) < y < f_{r+1}(x) = +\infty\}$, and any time elapsing for x_k keeps the valuation in C. □

Discrete successors. Since $\mathcal{D}_{\mathcal{A},\psi}$ is adapted in particular to *Poly*($\mathcal{A}$) which contains all guards, we have the following result:

Lemma 4.3. *Let $C \in \mathcal{D}_{\mathcal{A},\psi}$ be a cell of the aforementioned CAD. Let $v \in C$ be a valuation. Then for any $v' \in C$ and for every guard φ appearing in $\mathcal{A}$, $v' \models \varphi$ if, and only if, $v \models \varphi$.*

Hence we can write $C \models \varphi$ whenever $v \models \varphi$ and $v \in C$.

Moreover, for every update $x_i := P_i$ there is a polynomial $x_i - P_i$ in *Poly*($\mathcal{A}$), which has value 0 if and only if $x_i = P_i$; as a result:

Lemma 4.4. *Let $C \in \mathcal{D}^k_{\mathcal{A},\psi}$ be a cell of level k, $\underline{C}$ be the projection of C onto $\mathbb{R}^{k-1}$ and $-\infty = f_0 < \cdots < f_{r+1} = +\infty$ be the semi-algebraic functions dividing $\underline{C}$ as in Definition 3.2. Let u be an update of the form $x_k := P$ for some polynomial $P \in \mathbb{Q}[x_1, \ldots, x_{k-1}]$. Then there exists an index $i \in \{1, \ldots, r\}$ such that, over $\underline{C}$, $f_i = P$.*

As a corollary, there exists a unique cell $C' \in \mathcal{D}^k_{\mathcal{A},\psi}$ such that for any valuation $v \in C$, $v[u] \in C'$, namely $C' = \{(x, f_i(x)) \mid x \in \underline{C}\}$, which can be written $C[u]$.

Discrete transitions of $\mathcal{A}$ are translated as follows into $\mathcal{R}_{\mathcal{A},\psi}$: if $(q, \varphi, a, u, q') \in \Delta$ and $C \models \varphi$, there is a transition $(q, C) \xrightarrow{a} (q', C[u])$.

Proposition 4.5 (Correctness w.r.t. discrete steps).

- *If $(q, v) \xrightarrow{a} (q', v') \in \mathcal{T}_{\mathcal{A}}$, then $(q, C) \xrightarrow{a} (q', C') \in \mathcal{R}_{\mathcal{A}}$ with $v \in C$ and $v' \in C'$.*
- *If $(q, C) \xrightarrow{a} (q', C') \in \mathcal{R}_{\mathcal{A}}$ then for all $v \in C$ there exists $v' \in C'$ such that $(q, v) \xrightarrow{a} (q', v') \in \mathcal{T}_{\mathcal{A}}$.*

Proof.

- First, $(q, v) \xrightarrow{a} (q', v') \in \mathcal{T}_{\mathcal{A}}$ implies that there is a transition (q, φ, a, u, q') such that $v \models \varphi$ and $v' = v[u]$. By Lemma 4.3, we have that $C \models \varphi$. In addition, we have by Lemma 4.4 that $v' = v[u] \in C[u]$. By the definition of $\mathcal{R}_{\mathcal{A},\psi}$, there is a transition $(q, C) \xrightarrow{a} (q', C[u]) \in \mathcal{R}_{\mathcal{A},\psi}$.
- Transition $(q, C) \xrightarrow{a} (q', C') \in \mathcal{R}_{\mathcal{A},\psi}$ only exists because of a transition $(q, \varphi, a, u, q')\Delta$, and we have $C' = C[u]$. Let $v \in C$. Since $C \models \varphi$, by Lemma 4.3 we have that $v \models \varphi$. Hence there is a transition $(q, v) \xrightarrow{a} (q', v[u]) \in \mathcal{T}_{\mathcal{A}}$. By Lemma 4.4, $v[u] \in C[u]$, which concludes the proof. □

Example 4.6. Part of this abstraction for deciding reachability in PolITA $\mathcal{A}_0$ (Fig. 1, page 4) is depicted on Fig. 7. In this figure, points are given by the triangular system representing them. Computations of sample points for intervals between roots where omitted, and only appear in the graph as roots of derivatives. Note that having no a edge from state $q_0, 1, (5, \mathit{Int})$ is not an omission, but a consequence of the guard $x_1^2 \leq x_1 + 1$ no longer being satisfied. In this graph, C_+ is the polynomial obtained when replacing X_2 by $X_2 - 1$ in C. Faded states and transitions are unreachable but are nonetheless constructed from the decomposition.

Labeling with atomic propositions. Finally, we translate a comparison $P \bowtie 0$ in ψ into a fresh atomic proposition $p_{P \bowtie 0}$ and label $\mathcal{R}_{\mathcal{A},\psi}$ as follows. Note that since $\mathcal{D}_{\mathcal{A},\psi}$ is in particular adapted to $\mathit{Poly}(\psi)$, every cell C of $\mathcal{D}_{\mathcal{A},\psi}$ is sign-invariant for P, hence the truth value of $P \bowtie 0$ is constant in C. As a result, it makes sense to write $C \models P \bowtie 0$ whenever $v \models P \bowtie 0$ for some $v \in C$, and proposition $p_{P \bowtie 0}$ is true in every state (q, C) where $C \models P \bowtie 0$. We write $\overline{\psi}$ the formula where each $P \bowtie 0$ has been replaced by $p_{P \bowtie 0}$.

Proposition 4.7. *$\mathcal{A} \models \psi$ if, and only if, $\mathcal{R}_{\mathcal{A},\psi} \models \overline{\psi}$.*

Note that $\overline{\psi}$ is a CTL formula, which can be checked with the usual polynomial time labeling procedure. Since the number of cells in a cylindrical decomposition is doubly exponential in the number of clocks and polynomial in the number and maximal degree of polynomials to which it is adapted [6], we obtain the complexity stated in Theorem 4.1.

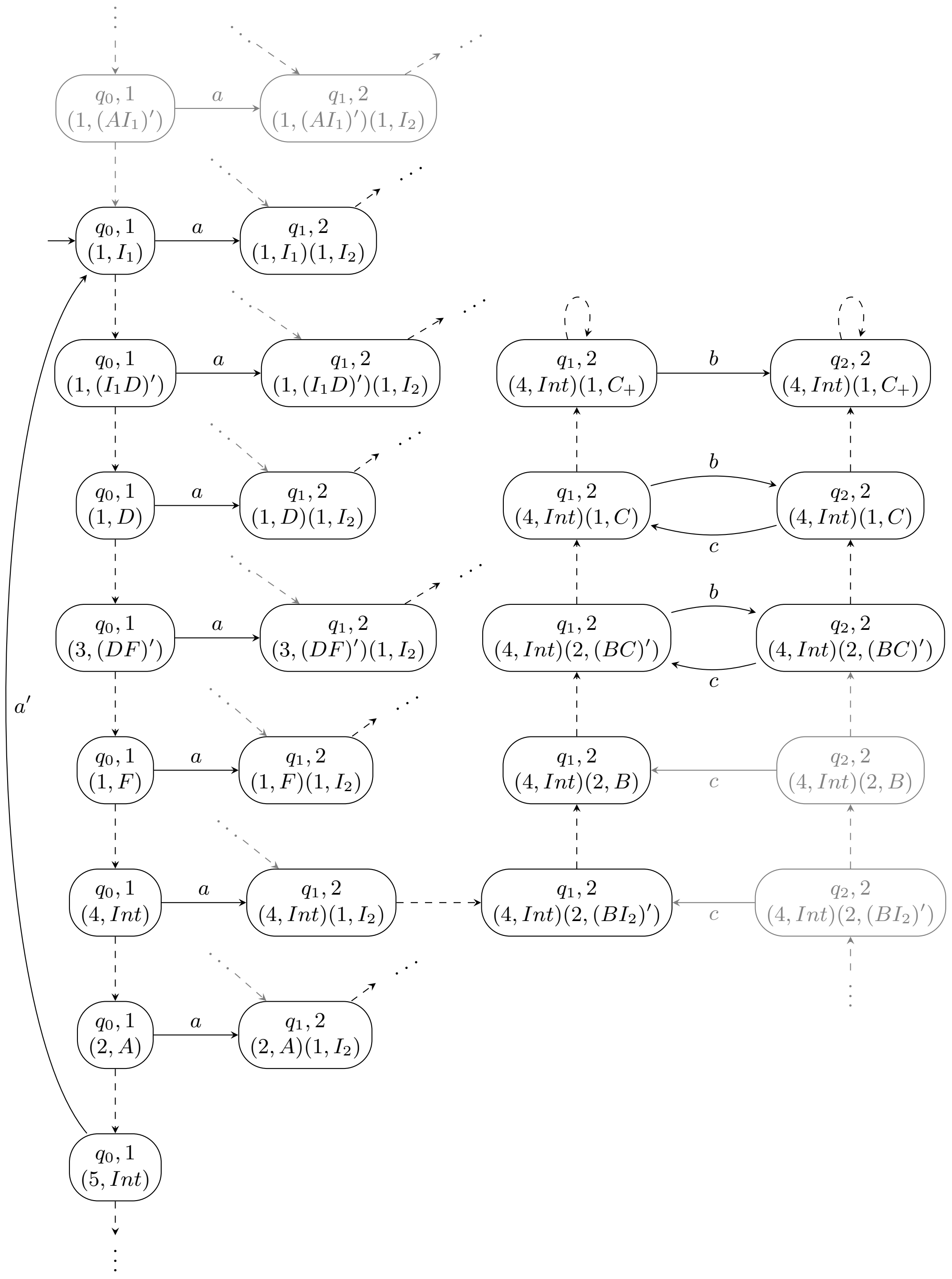

$q_0, 1$ $(1, (AI_1)')$
a
$q_1, 2$ $(1, (AI_1)')(1, I_2)$
$q_0, 1$ $(1, I_1)$
a
$q_1, 2$ $(1, I_1)(1, I_2)$
$q_0, 1$ $(1, (I_1D)')$
a
$q_1, 2$ $(1, (I_1D)')(1, I_2)$
$q_0, 1$ $(1, D)$
a
$q_1, 2$ $(1, D)(1, I_2)$
$q_0, 1$ $(3, (DF)')$
a
$q_1, 2$ $(3, (DF)')(1, I_2)$
$q_0, 1$ $(1, F)$
a
$q_1, 2$ $(1, F)(1, I_2)$
$q_0, 1$ $(4, Int)$
a
$q_1, 2$ $(4, Int)(1, I_2)$
$q_0, 1$ $(2, A)$
a
$q_1, 2$ $(2, A)(1, I_2)$
$q_0, 1$ $(5, Int)$
a'
$q_1, 2$ $(4, Int)(1, C_+)$
b
$q_2, 2$ $(4, Int)(1, C_+)$
$q_1, 2$ $(4, Int)(1, C)$
b
c
$q_2, 2$ $(4, Int)(1, C)$
$q_1, 2$ $(4, Int)(2, (BC)')$
b
c
$q_2, 2$ $(4, Int)(2, (BC)')$
$q_1, 2$ $(4, Int)(2, B)$
c
$q_2, 2$ $(4, Int)(2, B)$
$q_1, 2$ $(4, Int)(2, (BI_2)')$
c
$q_2, 2$ $(4, Int)(2, (BI_2)')$

Fig. 7. Partial depiction of $\mathcal{R}_{\mathcal{A}_0}$.
Dashed edges correspond to time successors *succ*; faded states are unreachable.

4.2 On-the-fly algorithm

Propositions 4.2 and 4.4 provide decidability of the model checking problem, by the algorithm that builds the finite graph $\mathcal{R}_{\mathcal{A},\psi}$ verifies that $\overline{\psi}$ is satisfied in this graph.

However, building the complete graph is not efficient in practice, since it requires to build the set of all cells beforehand. In the sequel, we show an on-the-fly algorithm that builds only the reachable part of $\mathcal{R}_{\mathcal{A},\psi}$. This algorithm would not, for example, build the faded states of $\mathcal{R}_{\mathcal{A}_0}$ in Fig. 7.

The key to the on-the-fly algorithm is to store only the part of the tree corresponding to the current sample point and its time successors. This construction is akin to what is done in Fig. 6, where only the line partitioning for X_2 above the current sample point is computed by the lifting phase, while line partitioning above, for, say, sample point $(1, F)$ is not computed. As a result, we do not keep the whole tree but only part of it.

We show that this information is sufficient to compute the successors through time elapsing and transition firing. Nonetheless, remark that although this pruning yields better performances in practice, the computational complexity in the worst case is not improved: the line partitioning at the first level already requires doubly exponential time, since the elimination phase is required.

Definition 4.8 (Pruned tree). *Let $\{\mathcal{P}_k\}_{k\leq n}$ be the polynomials obtained by the elimination phase. The* pruned tree *for sample point $(\alpha_1, \ldots, \alpha_k)$ is the sequence of completed line partitionings for sample points $\{(\alpha_1, \ldots, \alpha_i)\}_{1\leq i\leq k}$. By convention, the pruned tree for the empty sample point ($k = 0$) is the line partitioning at level* 1.

Given a clock valuation $(v_1, \ldots, v_k, 0, \ldots, 0)$ at level k, it can be represented by a sample point $(\alpha_1, \ldots, \alpha_k)$, or, equivalently, by a pruned tree for sample point $(\alpha_1, \ldots, \alpha_{k-1})$ and the index m of α_k in the line partitioning for $(\alpha_1, \ldots, \alpha_{k-1})$. In this representation, computing the time successors of $(\alpha_1, \ldots, \alpha_k)$ is simply done by incrementing m (if it is not the maximal index in the line partitioning). Note that in this algorithm we do not loop on the rightmost cell; although it is convenient to assume in $\mathcal{R}_{\mathcal{A}}$ that a time successor always exists, it has no effect regarding the reachability problem.

The set of enabled discrete transitions can be generated by computing the signs (see Algorithm 5 page 27) of polynomials appearing in guards. When a discrete transition $q \xrightarrow{g,a,u} q'$ is chosen, several cases should be distinguished with respect to the level of states q and q'.

– If the level decreases, *i.e.* $\lambda(q') < \lambda(q)$. Then the pruned tree corresponding to the new configuration is only the topmost-part of height $\lambda(q')$ of the original pruned tree. Otherwise said, we "forget" line partitionings for levels above $\lambda(q')$; however, the partitionings can be kept in memory in order not to have to recompute them later. The new index is the index of $\alpha_{\lambda(q')}$ in the partitioned line for this level.

- If the level doesn't change, *i.e.* $\lambda(q') = \lambda(q) = k$. The only way to change the clock values is through an update $x_k := P$ with $P \in \mathbb{Q}[X_1, \ldots, X_{k-1}]$. Then the polynomial of degree 1 $R = X_k - P$ was added to $Poly(\mathcal{A})$ and its unique root α'_k appears in the line partitioning of level ℓ. Note that in the triangular system representing $(\alpha_1, \ldots, \alpha'_k)$ it may appear as $\ldots(1, R)$ or some other equivalent value, hence to determine the index in the partitioned line the algorithm must actually determine the sign of R for all sample points of the line until 0 is found.
- If the level increases, *i.e.* $\lambda(q') > \lambda(q)$. First there can be an update of x_k, hence the same computations as above must be performed in order to find the new sample point corresponding to the valuation of clocks up to $\lambda(q)$. Then the pruned tree of height $\lambda(q')$ has to be computed. This is done by $\lambda(q') - \lambda(q)$ lifting steps (Algorithm 9 page 9). Since all clocks remain null for levels above $\lambda(q)$, the sample points given as input[4] are $(\alpha_1, \ldots, \alpha_{\lambda(q)}, 0, \ldots, 0)$.

Now the on-the-fly algorithm works as follows:

- Compute sets of polynomials $\{\mathcal{P}_i\}_{i \leq n}$ by the elimination phase.
- Compute the completed line partitioning at level 1.
- Start at a the initial state. If the level of the initial state is $k > 1$, proceed with $k - 1$ lifting phases as in the case of level increase. Add this state in a queue.
- Until the queue is empty:
  - Compute the list of fireable discrete transitions and whether time successor is allowed.
  - Add all new successors through a fireable discrete transition or a time step to the queue.
- Apply the model checking algorithm on this graph.

A note on efficient memory usage As noted above, a line partitioning only needs to be computed once. In addition – and this also holds for the complete construction of $\mathcal{R}_{\mathcal{A},\psi}$ –, the *triangular* structure of triangular systems enables a sharing of line partitioning at lower levels. Thus the size of the graph in memory is at most the size of the complete tree of the decomposition added, and not multiplied, by the number of states of the PolITA.

5 Expressiveness and extensions

We finally focus on expressiveness of PolITA. After comparing this class with stopwatch automata, we show how to extend it while keeping decidable the above

[4] Although the actual input of the algorithm are triangular systems, assuming we have the system $\mathcal{T}$ for $(\alpha_1, \ldots, \alpha_{\lambda(q)})$, the subsequent triangular systems are $\mathcal{T} \cup (1, X_{\lambda(q)+1}) \ldots$.

verification problems. For sake of clarity, in section 2 we have presented a basic model of PolITA. Here we show how to add three features consisting in: (1) including parameters in the expressions of guards and updates, (2) associating with each level a subset of auxiliary clocks, and (3) allowing to update clocks of lower levels than the current one. Since in the context of ITA, the first two extensions have already been studied in [9] and the third one in [10], our presentation will not be fully formalized.

5.1 PolITA *vs* Stopwatch automata

By syntax inclusion, PolITA are at least as expressive as ITA. As a direct consequence, there exists a timed language accepted by a PolITA that is not accepted by a TA [8].

There exists a timed language accepted by a timed automaton that is not accepted by any PolITA as presented above (the proof is a direct adaptation from the one proving said language is not accepted by an ITA [10]), although it is accepted by the extension with auxiliary clocks provided below (Section 5.3).

The class of stopwatch automata (SWA), which also syntactically contains the class of ITA, is however incomparable to PolITA.

Proposition 5.1. *There exists a timed language accepted by a* PolITA *with a single clock that cannot be accepted by a stopwatch automaton.*

The proof of the above proposition relies on a lemma about runs accepted by a SWA. Recall that in a stopwatch automaton, each clock can be active or inactive in every state. Also recall that updates are restricted to resets[5] $x := 0$ and guards are comparisons to a rational constant[6]. In the remainder of the section, we use $+_q$ to denote addition only on stopwatches active in q.

Lemma 5.2. *Let* $\rho = (q_0, v_0) \xrightarrow{\delta_0} (q_0, v_0 +_{q_0} \delta_0) \xrightarrow{g_0,a_0,u_0} (q_1, v_1) \cdots$ *be a run in a stopwatch automaton. Then there exists* $\rho' = (q_0, v_0) \xrightarrow{\delta'_0} (q_0, v_0 +_{q_0} \delta'_0) \xrightarrow{g,a_1,u} (q_1, v_1) \cdots$ *taking the same discrete transitions as* ρ *such that* $\forall i, \delta_i \in \mathbb{Q}$.

Proof. We assume that stopwatches are never reset throughout the run. This can be done since one can assume that a reset stopwatch is actually a fresh one. Consider the linear system with a variable δ_i per delay and rational coefficients which corresponds to all guards appearing after q_k. We write

$$\gamma_i^x = \begin{cases} 1 \text{ if } x \text{ is active in } q_i \\ 0 \text{ otherwise} \end{cases}$$

For each stopwatch x, we add the constraints

$$\bigwedge_{i=0}^{|\rho|} \left(\sum_{\ell=0}^{i} \gamma_i^x \cdot \delta_i \right) \models g_i$$

[5] It is possible to simulate affectations to rational constants, but it does not change expressiveness of the model.

[6] Again, diagonal constraints $x - y \bowtie c$ for $c \in \mathbb{Q}$ can be simulated.

Note that since guards have rational coefficients, this system has rational coefficients. In addition since ρ is an accepted run, this system has a solution $(\delta_0, \dots)$. Also note that for every solution $(\delta'_i)_i$, replacing each delay δ_i with δ'_i in ρ still yields a valid run ρ', since all guards are still respected. The set of solutions of a linear system with rational coefficient is a rational polyhedron, so the projection over each variable yields an interval with rational endpoints (or $-\infty$ or $+\infty$). If for some i, δ_i is irrational, the interval cannot be reduced to a point, so it contains an open set around δ_i, in which there is a rational δ'_i. Therefore, there exists a solution $(\delta'_i)_i \in \mathbb{Q}^{|\rho|}$ and ρ' is a run with rational delays. $\square$

Proof (Proposition 5.1). Consider PolITA of Fig. 8, which accepts the timed language $\mathcal{L}$ containing the single word $(a, 1)(b, \sqrt{2})$. Assume $\mathcal{L}$ is accepted by a stopwatch automaton $\mathcal{A}_\mathcal{L}$. Let $\rho = (q_0, v_0) \xrightarrow{\delta_0} (q_0, v_0 +_{q_0} \delta_0) \xrightarrow{g,a_1,u} (q_1, v_1) \cdots$ be a run accepting $(a, 1)(b, \sqrt{2})$. Note that some a_is may actually be ε. Since b occurs at an irrational instant, there is at least an irrational delay before the occurrence of b. By Lemma 5.2, ρ' the run where all delays are rational is also accepted. Therefore the instant of b in ρ' is rational and cannot be $\sqrt{2}$. Furthermore any time rescaling for $\mathcal{L}$ does not change this result since either a or b is taken at an irrational instant. $\square$

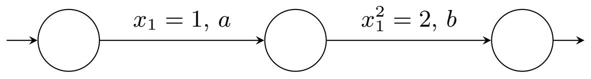


Fig. 8. A PolITA whose timed language is not accepted by a stopwatch automaton.

On the other hand, the (untimed) language of a PolITA (and the extensions of Section 5) is regular, as shown by the construction of a finite abstraction of $\mathcal{T}_\mathcal{A}$ in Section 4. It is not necessarily the case of (untimed) languages of stopwatch automata [12,2], hence there are some timed languages accepted by a SWA that are not accepted by any PolITA.

5.2 Parameters

Getting a complete knowledge of a system is often impossible, especially when integrating quantitative constraints. Moreover, even if these constraints are known, when the execution of the system slightly deviates from the expected behavior, due to implementation choices, previously established properties may not hold anymore. Additionally, considering a wide range of values for constants allows for a more flexible and robust design. Introducing parameters instead of concrete values is an elegant way of addressing these three issues. Parametrization however makes verification more difficult. For instance, in timed automata, allowing a single clock to be compared to parameters leads to undecidability of the reachability problem [21].

Suppose that we enlarge PolITA allowing expressions to be polynomials whose set of variables is the union of a set of clocks $\{x_1, \ldots, x_n\}$ and a set of parameters $\{p_1, \ldots, p_k\}$. Then we consider the cylindrical decomposition where the order of variables is $p_1, \ldots, p_k, x_1, \ldots, x_n$. Now assume that the relevant values of parameters are specified by a first-order formula *val*. Then using the cylindrical decomposition, we can answer reachability questions like "for all $p_1 \cdots p_k$ satisfying *val*, is q reachable?" or safety questions like "for all $p_1 \cdots p_k$ satisfying *val*, is q unreachable?".

5.3 Auxiliary clocks

With each level i, one may associate a set of *auxiliary* clocks Y_i in addition to the *main* clock x_i. Since there are multiple clocks for some level i, in this PolITA, with every state of level i, is associated an *active* clock among $X_i = \{x_i\} \cup Y_i$, specifying which clock evolves with time in this state. Auxiliary clocks may be used in a restrictive setting w.r.t. the main clocks to influence the behavior of the PolITA. Let us detail these restrictions:

- In a guard of a transition outgoing from a state at level i, among auxiliary clocks only those of the level i may occur and they are only be compared between them or with the main clock (i.e. $z \bowtie z'$ with $z, z' \in X_i$);
- In a transition outgoing from state at level i, an auxiliary clock of level i may be updated by another clock of level i (i.e. $y := z$ with $y \in Y_i$ and $z \in X_i$) while the main clock may be updated by an auxiliary clock only if the destination state of the transition is also at level i (i.e. $x_i := y$ with $y \in Y_i$).

The decision procedure works as follows. The cylindrical decomposition does not take into account the auxiliary clocks. However the definition of a class specifies in which interval of level i lies any clock of level i and their relative position for clocks inside the same interval.

Adding auxiliary clocks strictly extends expressiveness of PolITA w.r.t. timed languages. It was shown in [10] that the language

$$\begin{aligned} L = \big\{(a,t_1)(b,t_2) \ldots (a,t_{2p+1})(b,t_{2p+2}) \mid p \in \mathbb{N}, \\ \forall 0 \leq i \leq p,\ t_{2i+1} = i+1 \text{ and } i+1 < t_{2i+2} < i+2, \\ \forall 1 \leq i \leq p,\ t_{2i+2} - t_{2i+1} < t_{2i} - t_{2i-1}\big\} \end{aligned}$$

is not a language of an ITA. The proof also holds for PolITA since it is only based on the following hypotheses: (1) there is a single clock per level, (2) at level i, the behavior is only determined by the current state and the values of clocks at levels less or equal than i, and (3) the clock x_i is null at level $j < i$.

The untimed language of L is $(ab)^+$. In the accepted timed words, there is an occurrence of a at each time unit and the successive occurrences of b come each time closer to the next occurrence of a than previously. Consider the PolITA of Figure 9 with a single level and single final state q_2. The main clock x is active in all states and y is an auxiliary clock. It is routine to check that the timed language of this automaton is L.

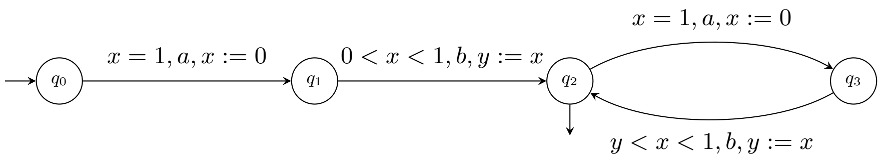


Fig. 9. A PolITA with a single level and an auxiliary clock

5.4 Allowing more updates

At level i, the value of a clock of level $j < i$ is relevant. So it is interesting to allow updates of such a clock. Again for keeping decidability, such updates have the following restrictions:

- At level i, the main clock of level $j < i$ can only be updated by a polynomial of the main clocks of level less than j: $x_j := P(x_1, \ldots, x_{j-1})$;
- At level i, an auxiliary clock of level $j < i$ may be updated by a clock of level j: $y := z$ with $y \in Y_j$ and $z \in X_j$.

The decision procedure for this extension consists in translating the extended PolITA in a PolITA with the same behavior by at level i: (1) delaying the update of clocks of level $j < i$ that should have been done until the current level becomes j and (2) duplicating the states by memorizing the current value of such a clock as an expression of the values of the clock when the level j was left. Guards and updates outgoing from a duplicated state are modified to take into account these expressions.

Let us illustrate this transformation on the PolITA of Figure 10 that is transformed in the PolITA of Figure 11. The original clock has only main clocks and the level of the state is indicated inside the state. In the transformed state the superscript '+' means that this corresponds to a state of of the original ITA ready to be simulated while the superscript '-' indicates that the delayed updates have to be performed. Let us start with the transition outgoing the state q_0, the update of x_1 is delayed but memorized in the state '$q_2^+, x_1 := 2$'. The transition outgoing from this state corresponds to the transition outgoing from q_2 but in the guard the occurrence of x_1 has been substituted by 2. With this transformation, the update becomes $x_2 := 5$ but since we are at level 3, this update is memorized in state '$q_3^+, x_1 := 2, x_2 := 5$'. The transition from q_3 at level 3 to q_5 at level 2 is split in two transitions in the simulating PolITA. First we enter state '$q_5^-, x_1 := 2, x_2 := 5$' at level 2 where the active clock is an auxiliary clock of level 2, y_2. Then in null time due to the guard we perform the delayed update of x_2, still memorizing the update of x_1 and enter the state '$q_5^+, x_1 := 2$'.

6 Conclusion

We extend Interrupt Timed Automata with polynomial expressions on clocks, and prove that reachability and model checking of some timed temporal logic

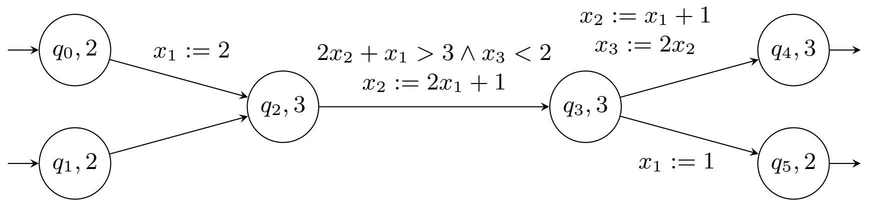
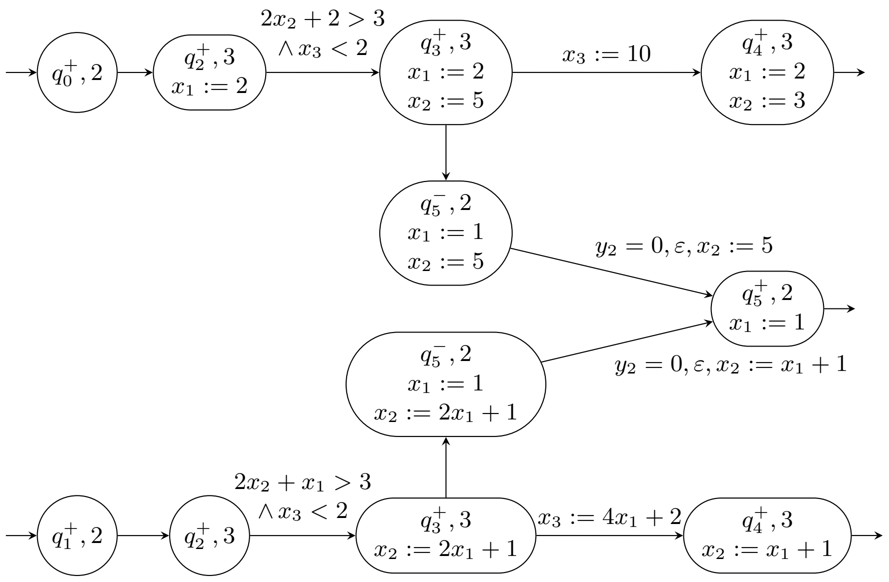


Fig. 10. A PoLITA containing extended updates of clocks



$q_0^+, 2$
$q_2^+, 3$ $x_1 := 2$
$2x_2 + 2 > 3$ $\wedge x_3 < 2$
$q_3^+, 3$ $x_1 := 2$ $x_2 := 5$
$x_3 := 10$
$q_4^+, 3$ $x_1 := 2$ $x_2 := 3$
$q_5^-, 2$ $x_1 := 1$ $x_2 := 5$
$y_2 = 0, \varepsilon, x_2 := 5$
$q_5^+, 2$ $x_1 := 1$
$q_5^-, 2$ $x_1 := 1$ $x_2 := 2x_1 + 1$
$y_2 = 0, \varepsilon, x_2 := x_1 + 1$
$q_1^+, 2$
$q_2^+, 3$
$2x_2 + x_1 > 3$ $\wedge x_3 < 2$
$q_3^+, 3$ $x_2 := 2x_1 + 1$
$x_3 := 4x_1 + 2$
$q_4^+, 3$ $x_2 := x_1 + 1$

Fig. 11. A PoLITA equivalent to the PoLITA of Figure 10

are decidable using the cylindrical decomposition. We also show that an on-the-fly construction of a class automaton is possible during the lifting phase of this decomposition. We establish that PoLITA and SWA are incomparable and provide some additional interesting features to the model. In order to experiment the practical complexity of the decision procedures, an implementation is in progress. Since the current construction still requires the full complexity of the cylindrical decomposition, we plan for future work to investigate if recent methods [14,19] with a lower complexity could be used to achieve reachability, possibly for a restricted version of PoLITA.

References

1. Alur, R., Courcoubetis, C., Dill, D.L.: Model-checking in dense real-time. Information and Computation 104, 2–34 (1993)

2. Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.H., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems. TCS 138, 3–34 (1995)
3. Alur, R., Dill, D.L.: A theory of timed automata. TCS 126, 183–235 (1994)
4. Alur, R., Henzinger, T.A., Lafferriere, G., Pappas, G.J.: Discrete abstractions of hybrid systems. Proceedings of the IEEE 88(7), 971–984 (2000)
5. Asarin, E., Maler, O., Pnueli, A.: Reachability analysis of dynamical systems having piecewise-constant derivatives. TCS 138(1), 35–65 (1995)
6. Basu, S., Pollack, R., Roy, M.F.: Algorithms in Real Algebraic Geometry. Springer (2006)
7. Ben-Or, M., Kozen, D., Reif, J.: The complexity of elementary algebra and geometry. In: Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing. pp. 457–464. STOC '84, ACM (1984)
8. Bérard, B., Haddad, S.: Interrupt timed automata. In: Proc. of FoSSaCS'09. LNCS, vol. 5504, pp. 197–211. Springer, York, UK (Mar 2009)
9. Bérard, B., Haddad, S., Jovanovič, A., Lime, D.: Parametric interrupt timed automata. In: Proceedings of the 7th Workshop on Reachability Problems in Computational Models (RP'13). LNCS, vol. 8169, pp. 59–69. Springer (2013)
10. Bérard, B., Haddad, S., Sassolas, M.: Interrupt timed automata: Verification and expressiveness. Formal Methods in System Design 40(1), 41–87 (Feb 2012)
11. Berman, L.: The complexity of logical theories. TCS 11(1), 71 – 77 (1980)
12. Cassez, F., Larsen, K.G.: The impressive power of stopwatches. In: Proc. of CONCUR'00. LNCS, vol. 1877, pp. 138–152. Springer (Aug 2000)
13. Collins, G.E.: Quantifier elimination for real closed fields by cylindrical algebraic decompostion. In: Automata Theory and Formal Languages 2nd GI Conference, LNCS, vol. 33, pp. 134–183. Springer Berlin Heidelberg (1975)
14. Din, M.S.E., Schost, E.: Polar varieties and computation of one point in each connected component of a smooth real algebraic set. In: Proceedings of International Symposium on Symbolic and Algebraic Computation (ISSAC 2003). pp. 224–231. ACM (2003)
15. Emerson, E.A., Halpern, J.Y.: Decision procedures and expressiveness in the temporal logic of branching time. In: Proc. 14th annual ACM Symp. on Theory of Computing (Stoc'82). pp. 169–180. ACM (1982)
16. Grossman, R., Nerode, A., Ravn, A., Rischel, H. (eds.): Hybrid systems, LNCS, vol. 736. Springer (1993)
17. Henzinger, T.A., Kopke, P.W., Puri, A., Varaiya, P.: What's decidable about hybrid automata? J. Comput. Syst. Sci. 57(1), 94–124 (1998)
18. Henzinger, T.A., Nicollin, X., Sifakis, J., Yovine, S.: Symbolic model checking for real-time systems. Information and Computation 111(2), 193–244 (1994)
19. Hong, H., Din, M.S.E.: Variant quantifier elimination. Journal of Symbolic Computation 47(7), 883–901 (2012)
20. Lafferriere, G., Pappas, G.J., Sastry, S.: O-minimal hybrid systems. MCSS 13(1), 1–21 (2000)
21. Miller, J.S.: Decidability and complexity results for timed automata and semi-linear hybrid automata. In: HSCC'00. LNCS, vol. 1790, pp. 296–309. Springer (2000)
22. Queille, J.P., Sifakis, J.: Specification and verification of concurrent systems in CESAR. In: Proceedings of the 5th International Symposium on Programming. LNCS, vol. 137, pp. 337–351. Springer-Verlag, London, UK (1982)
23. Tarski, A.: A decision method for elementary algebra and geometry. RAND Corporation, Santa Monica, Calif. (1948)