ARTICLE IN PRESS

Journal of Symbolic Computation ■ (■■■■) ■■■–■■■

Contents lists available at ScienceDirect

Journal of Symbolic Computation

journal homepage: www.elsevier.com/locate/jsc

Gröbner bases for polynomial systems with parameters

Antonio Montes [a,1], Michael Wibmer [b]

[a] Universitat Politècnica de Catalunya, Matematica Aplicada 2, c/ Jordi Girona 1-3, E–08034 Barcelona, Spain

[b] IWR, Universität Heidelberg, Germany

ARTICLE INFO

Article history:
Received 13 October 2009
Accepted 15 April 2010
Available online xxxx

Keywords:
Gröbner cover
Comprehensive
Reduced
Canonical
Parameters
Locally closed sets

ABSTRACT

Gröbner bases are the computational method par excellence for studying polynomial systems. In the case of parametric polynomial systems one has to determine the reduced Gröbner basis in dependence of the values of the parameters. In this article, we present the algorithm GRÖBNERCOVER which has as inputs a finite set of parametric polynomials, and outputs a finite partition of the parameter space into locally closed subsets together with polynomial data, from which the reduced Gröbner basis for a given parameter point can immediately be determined. The partition of the parameter space is intrinsic and particularly simple if the system is homogeneous.

© 2010 Elsevier Ltd. All rights reserved.

Introduction

Let K be a field and $\overline{K}$ be an algebraically closed extension of K (e.g. $K = \mathbb{Q}$ and $\overline{K} = \mathbb{C}$). A parametric polynomial system over K is given by a finite set of polynomials $p_1, \ldots, p_r \in K[\overline{a}, \overline{x}]$ in the variables $\overline{x} = x_1, \ldots, x_n$ and parameters $\overline{a} = a_1, \ldots, a_m$, and one is interested in studying the solutions of the algebraic systems $\{p_1(a, \overline{x}), \ldots, p_r(a, \overline{x})\} \subset \overline{K}[\overline{x}]$ which are obtained by specializing the parameters to concrete values $a \in \overline{K}^m$.

The computational approach par excellence for studying algebraic systems is the method of Gröbner bases and several articles have already been dedicated to the application of the ideas of Gröbner bases in the parametric setting, e.g. (Gianni, 1987; Weispfenning, 1992; Becker, 1994; Kapur, 1995; Duval, 1995; Kalkbrenner, 1997; Van Hentenryck et al., 1997; Moreno-Maza, 1997; Dellière, 1999; González-López et al., 2000; Gómez-Díaz, 2000; Fortuna et al., 2001; Montes, 2002; O'Halloran and Schilmoeller, 2002; Gao and Wang, 2003; Weispfenning, 2003; Sato and Suzuki, 2003; Sato, 2005;

E-mail addresses: antonio.montes@upc.edu (A. Montes), Wibmer@uni-heidelberg.de (M. Wibmer).
URL: http://www-ma2.upc.edu/~montes/ (A. Montes).
[1] Tel.: +34 934137704; fax: +34 934137701.

0747-7171/$ – see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jsc.2010.06.017

Please cite this article in press as: Montes, A., Wibmer, M., Gröbner bases for polynomial systems with parameters. Journal of Symbolic Computation (2010), doi:10.1016/j.jsc.2010.06.017

González-Vega et al., 2005; Nabeshima, 2005; Manubens and Montes, 2006; Nabeshima, 2006; Suzuki and Sato, 2006; Wibmer, 2007; Chen et al., 2007; Inoue et al., 2007; Inoue and Sato, 2007; Manubens, 2008; Manubens and Montes, 2009).

The first very important step was the proof of the existence of a *Comprehensive Gröbner Basis* together with an algorithm to obtain one via *Gröbner systems* in Weispfenning (1992). These algorithms have been implemented by Schönfeld (1991), Pesch (1994) and Dolzmann et al. (2006). To explain this fundamental concept we fix a monomial order $\succ_{\overline{x}}$ on the variables and an ideal $I \subset K[\overline{a}][\overline{x}] = K[\overline{a}, \overline{x}]$ (with generating set $\{p_1, \dots, p_r\}$). For $a \in \overline{K}^m$ we denote by $I_a \subset \overline{K}[\overline{x}]$ the ideal generated by all $p(a, \overline{x}) \in \overline{K}[\overline{x}]$ for $p \in I$.

A *Gröbner system* for I and $\succ_{\overline{x}}$ is a finite set of pairs $\{(S_1, B_1), \dots, (S_s, B_s)\}$ such that

(i) The S_i's are locally closed subsets of $\overline{K}^m$ such that $\overline{K}^m = \cup S_i$.
(ii) The B_i's are finite subsets of $K[\overline{a}][\overline{x}]$ and $B_i(a) = \{p(a, \overline{x}) : p \in B_i\}$ is a Gröbner basis of I_a with respect to $\succ_{\overline{x}}$ for every $a \in S_i$.
(iii) For $p \in B_i$ the function $a \mapsto \operatorname{lpp}(p(a, \overline{x}))$ is constant on S_i. In particular, $a \mapsto \operatorname{lpp}(I_a)$ is constant on S_i because of (ii), and so $\operatorname{lpp}(S_i) = \operatorname{lpp}(I_a)$ for some $a \in S_i$ is well-defined. (Here lpp denotes the leading power products with respect to $\succ_{\overline{x}}$.)

The S_i's are called the segments of the Gröbner system. Depending on the context one can also assume that the segments are arbitrary constructible subsets (as e.g. in Manubens and Montes (2009)), or locally closed subsets of the special form

$$\left\{a \in \overline{K}^m : f_1(a) = 0, \dots, f_s(a) = 0,\ g_1(a) \neq 0, \dots, g_t(a) \neq 0\right\} = \mathbb{V}(f_1, \dots, f_s) \setminus \mathbb{V}\left(\prod g_j\right)$$

with $f_1, \dots, f_s, g_1 \dots, g_t \in K[\overline{a}]$ as in Weispfenning (1992). In a more algorithmic context one usually replaces S_i with some polynomial data in the parameters that determines S_i. Some authors (e.g. Suzuki and Sato (2006)) also drop condition (iii). If one requires $B_i \subset I$ then the Gröbner system is called faithful. From a faithful Gröbner system one can obtain a comprehensive Gröbner bases B simply by $B = \cup B_i$. Our focus is on Gröbner systems rather than on comprehensive Gröbner bases because we think that the decomposition of the parameter space is very important in the applications.

After Weispfenning (1992), the effort has gone in two directions. Weispfenning (2003) and other authors (Manubens and Montes, 2009) worked in the direction of obtaining a canonical discussion only associated to the given ideal and monomial order, focusing on nice properties of the discussion. Other authors (Kapur, 1995; Kalkbrenner, 1997; Suzuki and Sato, 2006, 2007; Nabeshima, 2006) fixed their objective on effectiveness and speed.

A common problem with algorithms for the computation of Gröbner systems is that, mainly due to the large number of segments generated, the interpretation of the output can become quite tedious for the user.

Therefore the main focus of this article is not on the efficiency of the algorithm but on computing a Gröbner system that has as few segments as possible and satisfies some additional nice properties, so that the compact output allows an easy interpretation and the algorithm is easy to use in applications. Thus for us the crucial topic is how to actually represent all the reduced Gröbner bases for varying $a \in \overline{K}^m$ in the most simple and canonical way on the computer.

There is a certain difficulty with (reduced) Gröbner systems: Let $S \subset \overline{K}^m$ be a locally closed subset such that $a \mapsto \operatorname{lpp}(I_a)$ is constant on S and t an element of the minimal generating set of $\operatorname{lpp}(S)$. For $a \in S$ let $g(a)$ denote the element of the reduced Gröbner basis of I_a with $\operatorname{lpp}(g(a)) = t$. It is in general not possible to describe the function g on S by a single polynomial $p \in K[\overline{a}, \overline{x}]$. One reason for this can be that p might specialize to zero at a certain point $a \in S$, in other words, if we normalize p and consider it as element in $K(\overline{a})[\overline{x}]$ then $p(a, \overline{x})$ might not be defined for all $a \in S$ because some denominator specializes to zero. To avoid this kind of "singularities" we propose to use regular functions as in Wibmer (2007). We illustrate the above described phenomena with an example.

Example 1. Let $I = \langle ax + by, cx + dy\rangle \subset \mathbb{C}[a, b, c, d][x, y]$. We use a term-order with $x > y$. It is easy to see how the parameter space is partitioned according to lpp:

Please cite this article in press as: Montes, A., Wibmer, M., Gröbner bases for polynomial systems with parameters. Journal of Symbolic Computation (2010), doi:10.1016/j.jsc.2010.06.017

	Segment	lpp	Basis
S_1	$\mathbb{C}^4 \setminus \mathbb{V}(ad-bc)$	$\{y, x\}$	$\{y, x\}$
S_2	$\mathbb{V}(ad-bc) \setminus \mathbb{V}(a, c)$	$\{x\}$	$\{x + \left\{\frac{b}{a}, \frac{d}{c}\right\} y\}$
S_3	$\mathbb{V}(a, c) \setminus \mathbb{V}(a, b, c, d)$	$\{y\}$	$\{y\}$
S_4	$\mathbb{V}(a, b, c, d)$	$\{\}$	$\{\}$

There are four locally closed subsets of $\mathbb{C}^4$ with constant lpp. On S_2 neither $ax+by$ or $cx+dy$ alone is sufficient to describe the element of the reduced Gröbner bases of the specialized ideals because one of the leading coefficients always specializes to zero at a certain point in S_2. In fact the reader may convince himself that there does not exist a polynomial $p \in K[a, b, c, d][x, y]$ such that $p(\widetilde{a}, x, y)$ is a Gröbner basis of $I_{\widetilde{a}}$ for every $\widetilde{a} \in S_2$. This means that every Gröbner system necessarily decomposes S_2 in more than one segment (at least if (iii) is required or the Gröbner system is reduced). We think that it is undesirable to break up S_2 because intuitively the Gröbner basis structure of $I_{\widetilde{a}}$ is the same for every $\widetilde{a} \in S_2$ and we want to keep the number of segments as small as possible. Furthermore it seems that such a breaking up of S_2 can only be made in a canonical way if one uses some additional structure, like a term-oder in a, b, c, d as in Weispfenning (2003). We note that if $f : S_2 \to \mathbb{C}$ is the regular function given by $f(a, b, c, d) = \frac{b}{a}$ if $a \neq 0$ and $f(a, b, c, d) = \frac{d}{c}$ if $c \neq 0$ then the polynomial $x + fy \in \mathcal{O}(S_2)[\overline{x}]$ gives us precisely what we need.

In Weispfenning (2003) proposed a canonical form of comprehensive Gröbner bases along with a canonical Gröbner system. His recursive construction depends on an auxiliary well-quasi-order on the parameter ring $K[\overline{a}]$ and the number of segments in the canonical Gröbner system is not minimal. For example if $I = \langle ax, bx\rangle \subset \mathbb{C}[a, b][x]$ and we use the well-quasi-order on $\mathbb{C}[a, b]$ induced from the lexicographic order with $a > b$. Then the canonical Gröbner system is

$$\left\{\left(\mathbb{C}^2 \setminus \mathbb{V}(b), \{bx\}\right), (\mathbb{V}(b) \setminus \mathbb{V}(a, b), \{ax\}), (\mathbb{V}(a, b), \{\})\right\}.$$

Also the fact that the segments in the canonical Gröbner system are required to be irreducible causes more segments than strictly necessary and the segments need not be disjoint.

Here we will use a slightly different approach. We do not use bases B_i that are subsets of $K[\overline{a}][\overline{x}]$ but we use bases B_i that are subsets of $\mathcal{O}(S_i)[\overline{x}]$, where $\mathcal{O}(S_i)$ denotes the ring of regular functions on S_i. To emphasize this difference we call the resulting concept analogous to that of a Gröbner system a *Gröbner cover*. (See Section 1 for a precise definition.) In addition to the phenomena described in Example 1 the advantage of Gröbner covers is that the bases B_i are uniquely determined (by I, $\succ_{\overline{x}}$ and S_i). Even though this uniqueness is quite tautological we think it is preferable to have then an uniquely defined object of which we are computing a maybe non-unique representation than to have no uniqueness or only some weak kind of uniqueness (uniqueness under additional hypothesis) as in Weispfenning (2003).

Following Wibmer (2007) we also propose a canonical form of Gröbner covers. The precise definition of the *canonical Gröbner cover* is given in Section 1. The canonical Gröbner cover is uniquely determined by I and $\succ_{\overline{x}}$ and it is intrinsic in the sense that it does not depend on any algorithm. At all events if $I \subset K[\overline{a}][\overline{x}]$ is homogeneous with respect to the variables then the canonical Gröbner cover of $\overline{K}^m$ with respect to I and $\succ_{\overline{x}}$ is a set of pairs $\{(S_1, B_1), \ldots, (S_s, B_s)\}$ with the following properties:

(i) The S_i's are pairwise disjoint, locally closed subsets of $\overline{K}^m$ with $\overline{K}^m = \bigcup S_i$.
(ii) For $a, b \in \overline{K}^m$ we have $\mathrm{lpp}(I_a) = \mathrm{lpp}(I_b)$ if and only if there exists an i such that $a, b \in S_i$.
(iii) The B_i's are finite subsets of $\mathcal{O}(S_i)[\overline{x}]$ where $\mathcal{O}(S_i)$ denotes the ring of regular functions on S_i.
(iv) For $a \in S_i$ it holds that $\mathrm{lpp}(B_i)$ is the minimal generating set of $\mathrm{lpp}(I_a)$ and evaluating every element of B_i at $a \in S_i$ yields the reduced Gröbner basis of I_a with respect to $\succ_{\overline{x}}$.

In the above simple example the canonical Gröbner cover is

$$\left\{\left(\mathbb{C}^2 \setminus \mathbb{V}(a, b), x\right), (\mathbb{V}(a, b), \{\})\right\}.$$

The table in Example 1 also gives the canonical Gröbner cover. The canonical Gröbner cover has the nice property that it groups together *all* the values of the parameters for which the system of equations has the same type of solutions. This is in general not possible when one only uses polynomials instead of regular functions.

For non-homogeneous ideals a result as above is in general not obtainable (see Example 2 below), but using a process of homogenizing and dehomogenizing our algorithm GröbnerCover will give a similar result only that condition (ii) is not necessarily satisfied.

Example 2. Consider the non-homogeneous ideal $I = \langle ax+1, bx+1\rangle \subset \mathbb{C}[a,b][x]$. It is easy to see what we get if we somewhat inconsiderately simply partition the parameter space with respect to the lpp:

	Segment	lpp	Basis
1	$\left(\mathbb{C}^2 \setminus \mathbb{V}(a-b)\right) \cup \mathbb{V}(a,b)$	$\{1\}$	$\{1\}$
2	$\mathbb{V}(a-b) \setminus \mathbb{V}(a,b)$	$\{x\}$	$\{x+\frac{1}{a}\}$

The first segment with basis $\{1\}$ is not locally closed, i.e. it is not the difference of two closed sets. So condition (i) is not realized. But it is the union of the two disjoint locally closed sets $\mathbb{C}^2 \setminus \mathbb{V}(a-b)$ and $\mathbb{V}(a,b)$ and the reasons why we have basis $\{1\}$ over the point $\mathbb{V}(a,b)$ and why we have basis $\{1\}$ over $\mathbb{C}^2 \setminus \mathbb{V}(a-b)$ are fundamentally different. This difference can easily be detected using homogenization with respect to a new variable t. Homogenizing the system leads to $\langle ax+t, bx+t\rangle$. Now segment 1 splits into two segments 1a and 1b with distinct lpp as follows:

	Segment	lpp	Basis	Dehomogenized basis
1a	$\mathbb{C}^2 \setminus \mathbb{V}(a-b)$	$\{x,t\}$	$\{x,t\}$	$\{1\}$
1b	$\mathbb{V}(a,b)$	$\{t\}$	$\{t\}$	$\{1\}$
2	$\mathbb{V}(a-b) \setminus \mathbb{V}(a,b)$	$\{x\}$	$\{x+\frac{t}{a}\}$	$\{x+\frac{1}{a}\}$

Now all segments are locally closed but if we dehomogenize the segments 1a and 1b will of course have again the same lpp.

In 2006 Sato and Suzuki introduced a new very simple algorithm (Suzuki and Sato, 2006, 2007) to obtain a (comprehensive) Gröbner system. It seems very efficient in some problems but it is not a priori predicted which Gröbner system the algorithm will compute. Also the segments are not assumed to be disjoint and the algorithm might produce more segments than necessary. There are also concrete problems where these algorithms have been applied successfully (see e.g. González-López and Recio, 1993; Montes, 1998; Emiris, 1999; Rychlik, 2000; Yang et al., 2001; Coste, 2004; Montes and Recio, 2007).

The GröbnerCover algorithm is the outcome of a fruitful combination of the *Minimal Canonical Comprehensive Gröbner System* algorithm (Manubens and Montes, 2009) and the more theoretical results presented in Wibmer (2007). Depending on the point of view one can see this article as an intrinsic version of Manubens and Montes (2009) or an algorithmic version of Wibmer (2007).

Our algorithm has a long history (which is detailed in a series of papers of the first author (Montes, 2002; Manubens and Montes, 2006, 2009)), and many improvements have been made to fix the algorithms (see Manubens (2008)).

In fact, the GröbnerCover algorithm is the latest in a long line of algorithms (DisPGB, BuildTree, MCCGS) which have been introduced by the first author. The GröbnerCover algorithm uses some parts of these earlier algorithms. Since these parts are scattered over several articles and the results are not always present in exactly the way we would need it, we choose to give a new completely self-contained presentation. The very basic idea of the algorithm is still the same as in Weispfenning (1992) and the previous work of the first author. One uses a Buchberger like algorithm which branches whenever one needs to decide if a certain leading coefficient encountered in Buchberger's algorithm is zero or non-zero.

The essentially new contribution of this article starts when the Buchberger like algorithm BuildTree ends. New routines are LCUnion, Combine and Extend, as well as the method of homogenizing and dehomogenizing for non-homogeneous ideals that preserves the canonical character of the Gröbner cover. The global new thing is the complete algorithm that produces the Gröbner cover predicted in Wibmer (2007). Nevertheless we have also improved previous algorithms.

A critical point for a canonical description of a parametric ideal is the need of computing the radical of some sets of leading coefficients as was pointed out in Weispfenning (2003). Even the prime decomposition of these ideals in the parameters is needed. The first algorithm to compute prime decomposition of ideals was given in Gianni et al. (1988), and since there it has been improved. The interesting references for this are Giusti and Heintz (1990), Alonso and Raimondo (1990), Eisenbud et al. (1992) and Caboara et al. (1995). For further reading on the subject see Mora (2005) and references therein. It is known that this is a difficult problem (Heintz and Morgenstern, 1993), and so its use has been avoided by many authors. Nevertheless, in the discussion of parametric polynomial systems, the ideals in the parameters occurring in the computations are in general much simpler than the general ideals involved, and so the computation of prime decompositions is feasible. The algorithms involving radicals and primary decomposition are described in Section 2.1. There avoid the abusive use of primary decomposition. We also comment in Section 4.3 some details on how the routines involving radicals and primary decomposition should be implemented.

We now describe the content of the paper. Section 1 is purely theoretical and accurately defines the objects which will be computed in the subsequent sections. In particular the existence and uniqueness of a canonical partition of the parameter space is discussed. The main tool is a theorem for homogeneous ideals which, roughly speaking, states that in this case, the reduced Gröbner basis of I_a depends on a in an algebraic way as long as a is varied in subsets over which the lpp is constant. Most of the results of Section 1 have already been presented in Wibmer (2007) in a more general but maybe less accessible form.

In Section 2 we explain how the abstract concepts of Section 1 can be represented in a way feasible for computations. In 2.1 we first describe how we can represent locally closed sets. We introduce the *canonical representation* (C-representation) and the *canonical prime representation* (P-representation). Then, for the special locally closed sets used in BuildTree we introduce the *reduced representation* (R-representation). Then in 2.1.1 we describe the algorithm called *Locally Closed Union* (LCUnion) which computes the union of locally closed sets if their union is locally closed.

Then in the Sections 2.2 and 2.3 we explain how we represent regular and I-regular functions respectively and how we can effectively perform the corresponding operations. We introduce the full and the generic representation.

In Section 3 we describe the algorithm GCover, which is the heart of GröbnerCover algorithm. It computes the canonical Gröbner cover of a homogeneous ideal. After introducing some auxiliary algorithms (Section 3.1), we explain the BuildTree algorithm (Section 3.2) that yields a first disjoint reduced Gröbner System. Then GCover uses LCUnion to join together all the segments obtained by BuildTree with the same lpp to obtain the locally closed lpp-segments. Finally in 3.3 we describe the algorithm Basis that yields generic representations of the basis elements in the canonical Gröbner cover.

In Section 4 we present the main algorithm GröbnerCover. It distinguishes the two cases, whether the ideal under consideration is homogeneous or not. If it is not homogeneous the algorithm first homogenizes the ideal before calling GCover and then dehomogenizes, minimizes and reduces the bases in the output of GCover. At the end GröbnerCover converts the generic representations obtained by GCover into full representations. Finally, in Section 4.3, we make some comments about some strategies that can be used in practical problems and in the implementation.

In Section 5 we give an illustrative example.

The full GröbnerCover algorithm is currently being implemented in Singular and will be available freely.

1. Existence and uniqueness of the canonical partition of the parameter space

We first fix some notation which will be used throughout the paper: With K we denote a computable field and with $\overline{K}$ an algebraically closed field extension of K. (We do not insist that

Please cite this article in press as: Montes, A., Wibmer, M., Gröbner bases for polynomial systems with parameters. Journal of Symbolic Computation (2010), doi:10.1016/j.jsc.2010.06.017

$\overline{K}$ is the algebraic closure of K.) We fix $m, n \geq 1$ and an ideal $I \subset K[a_1, \ldots, a_m, x_1, \ldots, x_n] = K[\overline{a}, \overline{x}] = K[\overline{a}][\overline{x}]$. We call $\overline{a} = a_1, \ldots, a_m$ the *parameters* and $\overline{x} = x_1, \ldots, x_n$ the *variables*. We also fix a term-order $\succ_{\overline{x}}$ on the variables. If p is a polynomial in the variables with coefficients in some ring (e.g. $p \in K[\overline{a}][\overline{x}]$, $p \in \overline{K}[\overline{x}]$) then $\mathrm{lpp}(p)$ and $\mathrm{lc}(p)$ denote its leading power product (=leading monomial) and leading coefficient with respect to $\succ_{\overline{x}}$ respectively. A polynomial is called monic if its leading coefficient is equal to one.

The *parameter space* is $\overline{K}^m$. We consider it as a topological space by means of the K-Zariski topology. So a subset S of $\overline{K}^m$ is closed if and only if it is of the form

$$S = \mathbb{V}(N) := \left\{a \in \overline{K}^m : g(a) = 0 \;\forall\, g \in N\right\}$$

for some subset N of $K[\overline{a}] = K[a_1, \ldots, a_m]$.

If N is a subset of a ring we denote with $\langle N \rangle$ the ideal generated by N. For $N \subset K[\overline{a}]$ of course $\mathbb{V}(N) = \mathbb{V}(\langle N \rangle)$. If $\mathfrak{a}$ is an ideal of some ring then $\sqrt{\mathfrak{a}}$ denotes the radical of $\mathfrak{a}$.

Each point $a \in \overline{K}^m$ defines a morphism of K-algebras $\sigma_a : K[\overline{a}][\overline{x}] \to \overline{K}[\overline{x}]$ by sending the variables $\overline{x}$ to themselves and specializing the parameters with the concrete values given by a. We call σ_a the *specialization corresponding to a.*

Our goal is to describe the reduced Gröbner basis of $I_a := \langle \sigma_a(I) \rangle \subset \overline{K}[\overline{x}]$ (with respect to $\succ_{\overline{x}}$) in dependence of $a \in \overline{K}^m$.

We stress the point that although, for geometric purposes, we consider points $a \in \overline{K}^m$, on the algebraic side everything will be done over K (and not over $\overline{K}$). In particular all the polynomials we use have coefficients in K (and not in $\overline{K}$) and our algorithms (which will be detailed in the later sections) only use computations over K. Also it is important to notice that we always consider the K-Zariski topology on $\overline{K}^m$ (and never the $\overline{K}$-Zariski topology). We need to consider points in $\overline{K}^m$ to be able to use Hilbert's Nullstellensatz (Becker and Weispfenning, 1993, p. 313) which asserts that for every ideal $\mathfrak{a}$ of $K[\overline{a}]$

$$\mathbb{I}(\mathbb{V}(\mathfrak{a})) = \sqrt{\mathfrak{a}},$$

where for a subset V of $\overline{K}^m$ we define

$$\mathbb{I}(V) = \{g \in K[\overline{a}] : g(a) = 0 \text{ for all } a \in V\}.$$

From this it follows that $\mathbb{V}$ defines a bijection between the set of radical ideals of $K[\overline{a}]$ and the closed subsets of $\overline{K}^m$, the inverse mapping is given by $\mathbb{I}$. Under this bijection prime ideals correspond to irreducible closed subsets of $\overline{K}^m$ in the K-Zariski topology.

A subset S of $\overline{K}^m$ is called *locally closed* if it is open in its closure, or equivalently if it is the intersection of an open and a closed set. A function $f : S \to \overline{K}$ is called regular if for every $a \in S$ there exists an open neighborhood $U \subset S$ of a (i.e. $U = S \setminus V(M)$, with $a \in U$) such that

$$f(b) = \frac{p(b)}{q(b)} \quad \text{for all } b \in U$$

where $p, q \in K[\overline{a}]$ and $q(b) \neq 0$ for all $b \in U$. We denote the ring of regular functions on S by $\mathcal{O}(S)$.

Coarsely speaking, the ultimate goal of our algorithm GröbnerCover is to describe the function, that assigns to each $a \in \overline{K}^m$ the reduced Gröbner basis of I_a (with respect to $\succ_{\overline{x}}$) in "the most simple and natural way". Of course we will describe this function by using polynomials in some way or another and it seems reasonable to split $\overline{K}^m$ into segments S_i such that for all $a \in S_i$ the reduced Gröbner bases of I_a are of the same type, where we still need to make precise what we mean by "of the same type". It should mean firstly that $T := \mathrm{lpp}(I_a)$ does not depend on $a \in S_i$. As demonstrated in Example 2 (see also Example 3 in Wibmer (2007)) this first requirement is not enough and so we demand secondly that for each minimal generator t of T, the function that assigns to $a \in S_i$ the element of the reduced Gröbner basis of I_a with leading power product equal to t, depends on $a \in S_i$ in an algebraic way. The following two definitions make precise this idea.

Definition 3 (*I-regular Function*)**.** Let S be a locally closed subset of $\overline{K}^m$. We call a function $f : S \to \overline{K}[\overline{x}]$ *regular with respect to I* (or simply *I-regular* for short) if the following holds:

For each $a \in S$ there exists an open subset U of S with $a \in U$ and

$$f(b) = \frac{p(b, \overline{x})}{q(b)} \in \overline{K}[x] \quad \text{for all } b \in U, \tag{1}$$

where $p \in I$ and $q \in K[\overline{a}]$ such that $q(b) \neq 0$ for all $b \in U$.

The set of all I-regular functions on S is denoted by $\mathcal{I}(S)$. Obviously we can interpret $\mathcal{I}(S)$ as an ideal in the polynomial ring $\mathcal{O}(S)[\overline{x}]$. In particular the leading power product and leading coefficient is defined for an element of $\mathcal{I}(S)$. Intuitively we can think of $\mathcal{I}(S)$ as being the restriction of $I = \mathcal{I}(\overline{K}^m)$ to S.

Definition 4 (*Parametric Set*)**.** A locally closed subset S of $\overline{K}^m$ is called *parametric for I* (*with respect to $\succ_{\overline{x}}$*) if there exist monic I-regular functions $g_1, \ldots, g_r \in \mathcal{I}(S)$ such that $\{g_1(a), \ldots, g_r(a)\}$ is the reduced Gröbner basis of I_a for every $a \in S$.

From the uniqueness of reduced Gröbner bases it follows immediately that if $S \subset \overline{K}^m$ is parametric then the monic I-regular functions $g_1, \ldots, g_m \in \mathcal{I}(S)$ of Definition 4 are uniquely determined. We call them the *reduced Gröbner basis of I over S* (*with respect to $\succ_{\overline{x}}$*). Also the definition immediately implies that if a, b lie in a parametric set S then $\mathrm{lpp}(I_a) = \mathrm{lpp}(I_b)$. So we may define $\mathrm{lpp}(S) = \mathrm{lpp}(I_a)$ and call $\mathrm{lpp}(S)$ the *leading power products of I over S*.

The reader is referred to Wibmer (2007) for basic properties of parametric sets.

Remark 5. Let S be a locally closed subset of $\overline{K}^m$ such that $\mathrm{lpp}(I_a) = \mathrm{lpp}(I_b)$ for all $a, b \in S$ and let $t_1, \ldots, t_r$ be the minimal generating set of $\mathrm{lpp}(I_a) = \mathrm{lpp}(I_b)$. For each $i \in \{1, \ldots, r\}$ consider the function $g_i : S \to \overline{K}[\overline{x}]$ which sends $a \in S$ to the unique element of the reduced Gröbner basis of I_a with lpp equal to t_i. Then S is parametric if and only if for each $i = 1, \ldots, r$ the function g_i has the following natural property:

For each $a \in S$ there exists an open neighborhood U of a in S and a polynomial $p \in I$ such that $\mathrm{coef}(p, t_i)(b) \neq 0$ for all $b \in U$ and

$$g_i(b) = \frac{p(b, \overline{x})}{\mathrm{coef}(p, t_i)(b)} \in \overline{K}[\overline{x}]$$

for all $b \in U$.

Definition 6 (*Gröbner Cover*)**.** By a *Gröbner cover of $\overline{K}^m$ with respect to I and $\succ_{\overline{x}}$* we mean a finite set of pairs $\{(S_1, B_1), \ldots, (S_r, B_r)\}$ such that

- the S_i's are parametric and B_i is the reduced Gröbner basis of I over S_i for $i = 1, \ldots, r$ and
- the union of all S_i's equals $\overline{K}^m$.

The S_i's are called the segments of the Gröbner cover. The Gröbner cover is called disjoint if the S_i's are pairwise disjoint.

Our main algorithm GröbnerCover will compute a disjoint Gröbner cover of $\overline{K}^m$. But of course we want to specify a priori which Gröbner cover it will compute and surely this should be a particularly simple one. To give the definition of this unique canonical Gröbner cover the following theorem which was proved in Wibmer (2007) is essential.

Theorem 7. *Let $I \subset K[\overline{a}][\overline{x}]$ be a homogeneous ideal (with respect to the variables) and $a \in \overline{K}^m$. Then the set*

$$S := \left\{ b \in \overline{K}^m : \mathrm{lpp}(I_b) = \mathrm{lpp}(I_a) \right\}$$

is parametric. In particular S is locally closed.

From Theorem 7 the definition of the canonical Gröbner cover is quite obvious if I is a homogeneous ideal:

Theorem 8 (*Canonical Gröbner Cover*)**.** *If $I \subset K[\overline{a}][\overline{x}]$ is homogeneous (with respect to the variables) then there exists a unique Gröbner cover of $\overline{K}^m$ with minimal cardinality which we call the* canonical Gröbner cover of $\overline{K}^m$ (with respect to I and $\succ_{\overline{x}}$)*. It is disjoint and two points $a, b \in \overline{K}^m$ lie in the same segment if and only if* $\mathrm{lpp}(I_a) = \mathrm{lpp}(I_b)$*. The segments of this Gröbner cover will be called* lpp-segments*.*

1.1. The case of arbitrary ideals

For non-homogeneous ideals the situation is somewhat more complicated. But as we will see below we can use the method of homogenization to exploit Theorem 7 also in this case. For the rest of this section our focus is on the case that I is a non-homogeneous ideal. Our aim is to generalize the definition of the canonical Gröbner cover to arbitrary ideals.

For homogenization we introduce a new variable x_0 and extend $\succ_{\overline{x}}$ to the monomials in $x_0, x_1, \ldots, x_n$ by setting

$$\overline{x}^{\alpha} x_0^i \succ_{\overline{x},x_0} \overline{x}^{\beta} x_0^j$$

if $\overline{x}^{\alpha} \succ_{\overline{x}} \overline{x}^{\beta}$ or $\overline{x}^{\alpha} = \overline{x}^{\beta}$ and $i > j$. If p is a homogeneous polynomial in the variables $\overline{x}, x_0$ with coefficients in some ring then the *dehomogenization of* p is denoted with $\tau(p)$, i.e. $\tau(p) = p(\overline{x}, 1)$.

It is immediately seen that $\succ_{\overline{x},x_0}$ is a monomial order with the property that $\tau(\mathrm{lpp}(p)) = \mathrm{lpp}(\tau(p))$ and $\mathrm{lc}(\tau(p)) = \mathrm{lc}(p)$ for every *homogeneous* polynomial p.

Lemma 9 (*cf.* Eisenbud, 1994*, Exercise 15.39, p. 375*)**.** *Let $I' \subset K[\overline{x}]$ be an ideal and $J' \subset K[\overline{x}, x_0]$ a homogeneous ideal such that $\tau(J') = I'$. If $\{g_1, \ldots, g_r\}$ is a Gröbner basis of J' with respect to $\succ_{\overline{x},x_0}$ and the g_i's are homogeneous then $\{\tau(g_1), \ldots, \tau(g_r)\}$ is a Gröbner basis of I' with respect to $\succ_{\overline{x}}$.*

Proof. Let $p \in I'$. Then there exists $q \in J'$ homogeneous such that $\tau(q) = p$. Since $q \in J'$ there exists an i such that $\mathrm{lpp}(g_i)$ divides $\mathrm{lpp}(q)$, say $\mathrm{lpp}(q) = t\,\mathrm{lpp}(g_i)$. But then

$$\mathrm{lpp}(p) = \mathrm{lpp}(\tau(q)) = \tau(\mathrm{lpp}(q)) = \tau(t\,\mathrm{lpp}(g_i)) = \tau(t)\mathrm{lpp}(\tau(g_i)),$$

so that $\mathrm{lpp}(p)$ is divisible by $\mathrm{lpp}(\tau(g_i))$ and $\tau(g_1), \ldots, \tau(g_r)$ is a Gröbner basis of I'. □

Nevertheless it is not true that τ preserves reduced Gröbner bases. Consider, for example, the homogeneous ideal $F = \langle x^2y - yt^2 + t^3, x^2 - t^2 \rangle$. Its reduced Gröbner basis with respect to $\mathrm{grevlex}(x, y) \cdot \mathrm{lex}(t)$ is $G = \{t^3, x^2 - t^2\}$, that specializes to $G = \{1, x^2 - 1\}$ for $t = 1$. This is really a Gröbner basis of $F_{t=1}$ but not the reduced one which is $G_0 = \{1\}$.

Proposition 10. *Let $J \subset K[\overline{a}][\overline{x}, x_0]$ be a homogeneous ideal such that $\tau(J) = I$ and $S \subset \overline{K}^m$ parametric with respect to J and $\succ_{\overline{x},x_0}$. Then S is parametric with respect to I and $\succ_{\overline{x}}$.*

Proof. Let $h_1, \ldots, h_r \in \mathcal{O}(\underline{S})[\overline{x}, x_0]$ be the reduced Gröbner bases of J over S. We note that since J is homogeneous also $J_a \subset \overline{K}[\overline{x}, x_0]$ is homogeneous for every $a \in S$. The reduced Gröbner basis of a homogeneous ideal is homogeneous and so also $h_1, \ldots, h_r$ are homogeneous. Because $\mathrm{lc}(\tau(p)) = \mathrm{lc}(p)$ for homogeneous polynomials p we see that $\tau(h_1), \ldots, \tau(h_r) \in \mathcal{O}(S)[\overline{x}]$ are monic polynomials. Because $h_1, \ldots, h_r$ are J-regular, also $\tau(h_1), \ldots, \tau(h_r)$ are I-regular. Let $f_1, \ldots, f_s$ denote the monic I-regular functions obtained from $\tau(h_1), \ldots, \tau(h_r)$ by discarding those $\tau(h_i)$'s whose leading power product is divisible by some $\mathrm{lpp}(\tau(h_j))$ for $i \neq j$. Further let $g_1, \ldots, g_s \in \mathcal{O}(S)[\overline{x}]$ be the monic I-regular functions obtained by reducing f_i modulo $\{f_1, \ldots, f_s\} \setminus \{f_i\}$. To finish the proof we will show that $g_1(a), \ldots, g_s(a)$ is the reduced Gröbner basis of I_a for every $a \in S$. So choose $a \in S$. Since $h_1(a), \ldots, h_r(a)$ is Gröbner basis of J_a it follows from Lemma 9 that $\tau(h_1(a)), \ldots, \tau(h_r(a))$ is a Gröbner basis of $\tau(J_a) = I_a$. Therefore

$$\begin{aligned}\langle \mathrm{lpp}(I_a) \rangle &= \langle \mathrm{lpp}(\tau(h_1)(a)), \ldots, \mathrm{lpp}(\tau(h_r)(a)) \rangle = \langle \mathrm{lpp}(\tau(h_1)), \ldots, \mathrm{lpp}(\tau(h_r)) \rangle \\ &= \langle \mathrm{lpp}(f_1), \ldots, \mathrm{lpp}(f_s) \rangle = \langle \mathrm{lpp}(g_1), \ldots, \mathrm{lpp}(g_s) \rangle.\end{aligned}$$

This shows that $g_1(a), \ldots, g_s(a)$ is a Gröbner basis of I_a and since the g_i's are mutually reduced also the $g_i(a)$'s are mutually reduced. Consequently $g_1(a), \ldots, g_s(a)$ is the reduced Gröbner basis of I_a. □

Please cite this article in press as: Montes, A., Wibmer, M., Gröbner bases for polynomial systems with parameters. Journal of Symbolic Computation (2010), doi:10.1016/j.jsc.2010.06.017

Definition 11 (*Canonical Gröbner Cover*). Let $I \subset K[\overline{a}][\overline{x}]$ be an arbitrary ideal and let $J \subset K[\overline{a}][\overline{x}, x_0]$ denote its homogenization. By Proposition 10 the segments of the canonical Gröbner cover of $\overline{K}^m$ with respect to J and $\succ_{\overline{x},x_0}$ are parametric with respect to I and $\succ_{\overline{x}}$. The disjoint Gröbner cover of $\overline{K}^m$ with respect to I and $\succ_{\overline{x}}$ thus obtained will be called the *canonical Gröbner cover of $\overline{K}^m$ with respect to I and $\succ_{\overline{x}}$*.

In general homogenization does not commute with specialization. For example if we homogenize the polynomial $a_1x_1 + 1$ and then evaluate at $a_1 = 0$ we get x_0, but if we first evaluate and then homogenize we get 1. However, since the homogenization of a homogeneous polynomial is of course just the polynomial itself, there is no such problem if we only have to deal with homogeneous polynomials. So if $I \subset K[\overline{a}][\overline{x}]$ already was homogeneous we immediately see that for $a \in \overline{K}^m$ the reduced Gröbner basis of J_a with respect to $\succ_{\overline{x},x_0}$ equals the reduced Gröbner basis of I_a with respect to $\succ_{\overline{x}}$. From this observation it follows that the definition of the canonical Gröbner cover is unambiguous. I.e. if the ideal I in Definition 11 is already homogeneous then Definition 11 agrees with the definition in Theorem 8.

2. Representations and some associated computations

In Section 1 we defined the canonical Gröbner cover (see Theorem 8 and Definition 11). But before explaining the algorithm to compute this object, we need to know how we can actually represent all the objects (locally closed sets, regular functions, I-regular functions) appearing in the definitions. And we also need to be able to perform the evident operations (e.g. boolean combinations of locally closed sets, addition and multiplication of regular functions, reduction modulo I-regular functions) with this representations. This is the objective of this second chapter.

2.1. Representation of locally closed sets

In this section we introduce the canonical representation (C-representation) and the prime representation (P-representation) of locally closed sets. We also present the reduced representation (R-representation) which only applies to a special class of locally closed sets which will be used during the BuildTree algorithm. We recall that a subset $S \subseteq \overline{K}^m$ is called *locally closed* if it is open in its closure. This is equivalent to saying that S is of the form $S = \mathbb{V}(\mathfrak{a}) \setminus \mathbb{V}(\mathfrak{b})$ for subsets $\mathfrak{a}$, $\mathfrak{b}$ of $K[\overline{a}]$.

Definition 12 (*C-representation*). Let $S \subset \overline{K}^m$ be a locally closed set. There exist uniquely determined radical ideals $\mathfrak{a}$, $\mathfrak{b}$ of $K[\overline{a}]$, with $S = \mathbb{V}(\mathfrak{a}) \setminus \mathbb{V}(\mathfrak{b})$ and $\mathfrak{a} \subset \mathfrak{b}$, such that

- $\overline{S} = \mathbb{V}(\mathfrak{a})$ and
- $\overline{S} \setminus S = \mathbb{V}(\mathfrak{b})$.

The pair $(\mathfrak{a}, \mathfrak{b})$ is called the *C-representation* of S.

Proof. Since S is open in $\overline{S}$ we see that $\overline{S} \setminus S$ is closed. Existence and uniqueness now follows from the one to one correspondence between closed sets and radical ideals. □

Remark 13. A locally closed set is closed if and only if $\mathfrak{b} = \langle 1 \rangle$.

Definition 14 (*P-representation*). Let $S \subset \overline{K}^m$ be a locally closed set. There exists uniquely determined prime ideals

$$\{(\mathfrak{p}_i, \{\mathfrak{p}_{ij} : 1 \leq j \leq r_i\}) : 1 \leq i \leq r\} \tag{2}$$

of $K[\overline{a}]$, with $S = \bigcup_i \left(V(\mathfrak{p}_i) \setminus \bigcup_j \mathbb{V}(\mathfrak{p}_{ij})\right)$ and $\mathfrak{p}_i \subset \mathfrak{p}_{ij}$ for all i, j, such that

- $\overline{S} = \mathbb{V}(\mathfrak{p}_1) \cup \cdots \cup \mathbb{V}(\mathfrak{p}_r)$ and
- $(\overline{S} \setminus S) \cap \mathbb{V}(\mathfrak{p}_i) = \mathbb{V}(\mathfrak{p}_{i1}) \cup \cdots \cup \mathbb{V}(\mathfrak{p}_{ir_i})$

are the minimal decompositions into irreducible closed sets. We call (2) the *P-representation of* S. The $\mathfrak{p}_i$'s are called the *components of* S and the $\mathfrak{p}_{ij}$ are called the *holes of* $\mathfrak{p}_i$ *(with respect to* S*)*.

Table 1
RrepNN algorithm.

```
(𝔞′, h′) ← RrepNN(𝔞, h, f)
Input:
    (𝔞, h) an R-representation
    f ∈ K[ā] assumed to be non-null on the restriction of S = 𝕍(𝔞) ∖ 𝕍(h).
Output:
    (𝔞′, h′): the R-representation of S1 = 𝕍(𝔞) ∖ 𝕍(hf)

begin
  h1 := hf
  𝔞′ := 𝔞 : ⟨h1⟩
  h′ := squarefree(h1)
end[a]
```

[a] In practical implementation h_1 should be reduced modulo $\mathfrak{a}$ and $\mathfrak{a}'$.

Proof. Since $\overline{S} \setminus S$ is closed the existence and uniqueness follows from the existence and uniqueness of the minimal decomposition of a closed set into irreducible closed sets. □

In the first step BuildTree of the GröbnerCover algorithm, appear a special kind of locally closed sets for which the following definition and representation is needed.

Definition 15 (*R-representation*)**.** Let $S \subset \overline{K}^m$ be a locally closed subset of the form

$$S = S((\mathfrak{a}, h)) = \mathbb{V}(\mathfrak{a}) \setminus \mathbb{V}(h),$$

where $\mathfrak{a} \subset K[\overline{a}]$ is an ideal and $h \in K[\overline{a}]$. We say that the pair $(\mathfrak{a}, h)$ is an R-representation of S if

- $\mathfrak{a}$ is radical,
- $\overline{S} = \mathbb{V}(\mathfrak{a})$,
- h is square-free (radical).[2]

Remark 16. For a locally closed set allowing an R-representation, the ideal $\mathfrak{a}$ in the R-representation is the same as in the C-representation, but the polynomial h is not unique. For example consider the locally closed set S defined by the R-representation $(\langle a - b^2\rangle, a^2 - b)$. It is easy to see that $(\langle a - b^2\rangle, b(b-1)(b^2+b+1))$ is also a (better) R-representation representing S.

Proposition 17. *Let $(\mathfrak{a}, h)$ be an R-representation of the locally closed set S, and let $f \in K[\overline{a}]$ be such that $f \notin \mathfrak{a}$. Then, the algorithm* RrepNN *of Table 1 computes an R-representation of the locally closed set $S_1 = \mathbb{V}(\mathfrak{a}) \setminus \mathbb{V}(hf)$.*

Proof. We can decompose the proof in simpler steps. Let $\mathfrak{a}, \mathfrak{b}, \mathfrak{p}$ be ideals of $K[\overline{a}]$ and $g \in K[\overline{a}]$. Then

(a) If $g \in \mathfrak{b}$ then $\mathfrak{b} : \langle g\rangle = \langle 1\rangle$.
(b) If $\mathfrak{p}$ is prime and $g \notin \mathfrak{p}$ then $\mathfrak{p} : \langle g\rangle = \mathfrak{p}$.
(c) If $\mathfrak{a}$ is radical and $\mathfrak{a} = \cap_i \mathfrak{p}_i$ is its prime decomposition then $\mathfrak{a} : \langle h\rangle = \bigcap_{h \notin \mathfrak{p}_i} \mathfrak{p}_i = \mathfrak{a}'$ (also radical).
(d) If $\mathfrak{a}$ is radical and $S = \mathbb{V}(\mathfrak{a}) \setminus \mathbb{V}(h)$ then $\overline{S} = \mathbb{V}(\mathfrak{a} : \langle h\rangle)$. Thus setting $\mathfrak{a}' = \mathfrak{a} : \langle h\rangle$ the R-representation of S is $(\mathfrak{a}', h)$.

Proposition 17 follows from (d). We let the proofs as an exercise. □

Proposition 18. *Let $(\mathfrak{a}, h)$ be an R-representation of the locally closed set S, and let $f \in K[\overline{a}]$ be such that $f \notin \mathfrak{a}$. Then, the algorithm* RrepN *of Table 2 computes an R-representation of the locally closed set $S_0 = \mathbb{V}(\mathfrak{a} + \langle f\rangle) \setminus \mathbb{V}(h)$.*

[2] In practical implementation h should be reduced modulo $\mathfrak{a}$, but this is not needed for theoretical purposes.

ARTICLE IN PRESS

Table 2
RrepN algorithm.

```
(𝔞′, h′) ← RrepN(𝔞, h, f)
Input:
      (𝔞, h) an R-representation
      f ∈ K[ā] assumed to be null on the restriction of S = 𝕍(𝔞) ∖ 𝕍(h).
Output:
      (𝔞′, h): the R-representation of S_0 = 𝕍(𝔞 + ⟨f⟩) ∖ 𝕍(h)

begin
   𝔞_1 := √(𝔞 + ⟨f⟩)
   𝔞′ := 𝔞_1 : ⟨h⟩
end
```

Proof. The proof of Proposition 18 follows as the proof of Proposition 17, and we let it as an exercise. □

The usefulness of reduced representations comes from the following

Proposition 19 (*Split*). *Let* $(\mathfrak{a}, h)$ *be the R-representation of the locally closed set* $S = \mathbb{V}(\mathfrak{a}) \setminus \mathbb{V}(h) \subset \overline{K}^m$ *and* $f \in K[\overline{a}]$. *Then*

(i) $f(a) = 0$ *for all* $a \in S$ *if and only if* $f \in \mathfrak{a}$.
(ii) $f(a) \neq 0$ *for all* $a \in S$ *if and only if* $\text{RrepN}(\mathfrak{a}, h, f) = (\langle 1 \rangle, h')$.
(iii) *If neither* $f(a) = 0$ *nor* $f(a) \neq 0$ *holds for all* $a \in S$ *then S is the disjoint union of the two non-empty disjoint locally closed sets*

$$S_0 = S\,(\text{RrepN}(\mathfrak{a}, h, f)) \quad \textit{and} \quad S_1 = S\,(\text{RrepNN}(\mathfrak{a}, h, f))$$

and $f(a) = 0$ *for all* $a \in S_0$ *whereas* $f(a) \neq 0$ *for all* $a \in S_1$.
(iv) *If* $f \notin \mathfrak{a}$ *then the algorithm* Split *in Table 3 outputs two new R-representations* $(\mathfrak{a}_0, h_0)$ *and* $(\mathfrak{a}_1, h_1)$ *that splits S into two disjoint sets* $S_0 = \mathbb{V}(\mathfrak{a}_0) \setminus \mathbb{V}(h_0)$ *and* $S_1 = \mathbb{V}(\mathfrak{a}_1) \setminus \mathbb{V}(h_1)$ *such that*
  – $S_0 \cup S_1 = S$ *and* $S_0 \cap S_1 = \emptyset$,
  – $f(a) = 0$ *for all* $a \in S_0$ *and* $f(a) \neq 0$ *for all* $a \in S_1$,
  – $\mathfrak{a}_0 = \langle 1 \rangle$ *if and only if* $S_0 = \emptyset$, *so that no splitting is necessary.*

Proof.

(i) Obviously, if $f \in \mathfrak{a}$ then $f(a) = 0$ for all $a \in S((\mathfrak{a}, h))$. For the reciprocal, if $f(a) = 0$ for all $a \in S((\mathfrak{a}, h))$ then f also vanishes on the closure $\overline{S((\mathfrak{a}, h))} = \mathbb{V}(\mathfrak{a})$. Thus, as $\mathfrak{a}$ is radical, by Hilbert's Nullstellensatz it follows that $f \in \mathfrak{a}$.
(ii) The set of all points of $S = \mathbb{V}(\mathfrak{a}) \setminus \mathbb{V}(h)$ where f vanishes is $\mathbb{V}(\mathfrak{a} + \langle f \rangle) \setminus \mathbb{V}(h)$. Thus $f(a) \neq 0$ for all $a \in S((\mathfrak{a}, h))$ if and only if $\mathbb{V}(\mathfrak{a} + \langle f \rangle) \setminus \mathbb{V}(h) = \emptyset$ and this is equivalent to $\text{RrepN}(\mathfrak{a}, h, f) = (\langle 1 \rangle, 1)$.
(iii) Obvious from Definition 15.
(iv) Follows from (iii). □

As it is described later, GröbnerCover builds the first Gröbner system using BuildTree that uses R-representations, but when it finishes one needs to transform them into P-representations. The algorithm RtoPrep in Table 4 will do it. It uses the PrimeDecomp algorithm. PrimeDecomp computes the minimal prime ideals of the radical of a given ideal of $K[\overline{a}]$ (see Gianni et al., 1988; Mora, 2005). We have already commented in the Introduction the complexity of the prime decomposition (Heintz and Morgenstern, 1993). Nevertheless it should be noted that RtoPrep needs only 2 special types of prime decompositions. In the first one, we already know that the given ideal is radical, and in the second we compute the prime decomposition of a prime ideal plus a square-free polynomial (and non-reducible modulo the prime ideal). These operations are simpler as the general prime decomposition, and special algorithms for this should be designed.

A further observation is that the ideals involved in parametric polynomial discussions are usually not very complex and so the operations involved are not so time consuming.

Please cite this article in press as: Montes, A., Wibmer, M., Gröbner bases for polynomial systems with parameters. Journal of Symbolic Computation (2010), doi:10.1016/j.jsc.2010.06.017

ARTICLE IN PRESS

Table 3
SPLIT algorithm.

```
((a_0, h_0), (a_1, h_1)) ← Split(f, (a, h))
Input:
      (a, h): an R-representation of S = V(a) \ V(h)
      f ∈ K[ā]: a new polynomial not in a
Output:
      (a_0, h_0): R-representation of the points a ∈ S with f(a) = 0
      (a_1, h_1): R-representation of the points a ∈ S with f(a) ≠ 0

begin
   (a_0, h_0) := RREPN(a, h, f)
   if a_0 = ⟨1⟩ then
      (a_1, h_1) := (a, h)
   else
      (a_1, h_1) := RREPNN(a, h, f)
   end if
end
```

Table 4
RTOPREP algorithm.

```
T ← RtoPrep(a, h)
Input:
      (a, h) an R-representation.
Output:
      T = {(p_i, {p_ij : 1 ≤ j ≤ s_i}) : 1 ≤ i ≤ s}: the P-representation of
          V(a) \ V(h)

begin
   T := ∅
   D := PRIMEDECOMP(a)
   for p ∈ D do
      T_p := PRIMEDECOMP(p + ⟨h⟩)
      T := T ∪ {p, T_p}
   end for
end
```

2.1.1. Computing the union of locally closed sets

Let $S_1, \dots, S_r$ be locally closed subsets of $\overline{K}^m$. In this subsection we present the algorithm LCUNION (see Table 5) which computes their union $S = S_1 \cup \dots \cup S_r$ under the assumptions that S is locally closed and the S_i's are pairwise disjoint. In our main algorithm GRÖBNERCOVER such a situation will occur when BUILDTREE has finished, because of Theorem 7. The computational aspects of boolean operations with locally closed sets have already been treated in the literature (see e.g. O'Halloran and Schilmoeller (2002), Chen et al. (2009) and Manubens and Montes (2009)). But in general the union of locally closed sets need not be locally closed and the above mentioned two assumptions can be used to significantly simplify and speed up the computation.

The first while loop in ADDPART is present for efficiency reasons as it will do, in a simple way, "most of the work", but the true algorithm is the second while loop. These routines use SELECTMINIDEALS that from a set of prime ideals selects the minimal ideals that do not contain each others.

Proposition 20. *Let $S_1, \dots, S_r$ be pairwise disjoint, locally closed subsets of $\overline{K}^m$ such that their union $S = S_1 \cup \dots \cup S_r$ is locally closed. Then* LCUNION *computes the P-representation of S.*

Proof. As in the algorithm we assume that S_i is given in the P-representation

$$\left\{\left(\mathfrak{p}^i_j, \{\mathfrak{p}^i_{jk} : 1 \le k \le r^i_j\}\right) : 1 \le j \le r^i\right\}.$$

Since $\overline{S} = \overline{S_1} \cup \dots \cup \overline{S_r} = \mathbb{V}(\cap_{i,j}\mathfrak{p}^i_j)$ it is clear that the minimal elements of the set $\left\{\mathfrak{p}^i_j : 1 \le i \le r, 1 \le j \le r_j\right\}$ are the components of S. Therefore we already see that LCUNION yields the correct

Please cite this article in press as: Montes, A., Wibmer, M., Gröbner bases for polynomial systems with parameters. Journal of Symbolic Computation (2010), doi:10.1016/j.jsc.2010.06.017

ARTICLE IN PRESS

Table 5
LCUnion algorithm.

$T \leftarrow$ **LCUnion**$(S_1, \dots, S_r)$
Input:
$S_1, \dots, S_r$; pairwise disjoint locally closed subsets of $\overline{K}^m$ such that their union is locally closed
Output:
The P-representation of $S = S_1 \cup \cdots \cup S_r$

begin
Assume that the S_i's are given in the P-representation

$$\left\{\left(\mathfrak{p}_j^i, \{\mathfrak{p}_{jk}^i : 1 \le k \le r_j^i\}\right) : 1 \le j \le r^i\right\}.$$

$P := \text{SelectMinIdeals}\left(\{\mathfrak{p}_j^i : 1 \le i \le r, 1 \le j \le r^i\}\right)$
$T := \emptyset$
for $\mathfrak{p} \in P$ **do**
Let $H = \{\mathfrak{q}_1, \dots, \mathfrak{q}_s\}$ be the holes of $\mathfrak{p}$.

$$C := \left\{\left(\mathfrak{p}_j^i, \{\mathfrak{p}_{j1}^i, \dots, \mathfrak{p}_{jr_j^i}^i\}\right) : 1 \le i \le r, \ \mathfrak{p} \text{ is not a component of } S_i, \text{ for } 1 \le j \le r^i\right\}$$

$$T := T \cup \left\{(\mathfrak{p}, \text{AddPart}(H, C))\right\}$$

end for
end

components. It remains to prove that LCUnion yields the correct holes. For this we fix a component $\mathfrak{p}$ of S. As seen above $\mathfrak{p}$ is also a component of some S_{i_0}. Since the S_i's are pairwise disjoint this S_{i_0} is uniquely determined. We have to show that algorithm AddPart transforms the holes $H = \{\mathfrak{q}_1, \dots, \mathfrak{q}_s\}$ of $\mathfrak{p}$ with respect to S_{i_0} into the holes of $\mathfrak{p}$ with respect to S. More precisely, let as in the algorithm C be the set whose elements are of the form $\left(\mathfrak{p}_j^i, \{\mathfrak{p}_{j1}^i, \dots, \mathfrak{p}_{jr_j^i}^i\}\right)$ with $i \in \{1, \dots, r\} \setminus \{i_0\}$ and $1 \le j \le r^i$ and $\{\mathfrak{q}'_1, \dots, \mathfrak{q}'_{s'}\} = \text{AddPart}(H, C)$. Then, according to Definition 14 we have to show that $\mathbb{V}(\mathfrak{q}'_1) \cup \cdots \cup \mathbb{V}(\mathfrak{q}'_{s'})$ is the minimal decomposition of $(\overline{S} \setminus S) \cap \mathbb{V}(\mathfrak{p})$ into irreducible closed sets. Because of the usage of SelectMinIdeals there are no inclusions between the $\mathfrak{q}'_j$'s and therefore it suffices to show that

$$(\overline{S} \setminus S) \cap \mathbb{V}(\mathfrak{p}) = \mathbb{V}(\mathfrak{q}'_1) \cup \cdots \cup \mathbb{V}(\mathfrak{q}'_{s'}). \tag{3}$$

During algorithm AddPart the set Q of prime ideals gets modified in every step. When a new element, say $\mathfrak{p}'$, is being added to Q then it always satisfies $\mathfrak{p}' \supsetneqq \mathfrak{q}$ for some $\mathfrak{q}$ which is being deleted from Q. In particular in every step the closed set $\cup_{\mathfrak{q} \in Q} \mathbb{V}(\mathfrak{q})$ gets strictly smaller. This shows that AddPart will terminate. Also as we have $\mathfrak{p} \subsetneqq \mathfrak{q}_i$ for the "initial" holes $\{\mathfrak{q}_1, \dots, \mathfrak{q}_s\}$ of $\mathfrak{p}$ with respect to S_{i_0} we obtain $\mathfrak{p} \subsetneqq \mathfrak{q}$ for every $\mathfrak{q} \in Q$ in every step. In particular $\mathbb{V}(\mathfrak{q}'_i) \subset \mathbb{V}(\mathfrak{p})$ for $i = 1, \dots, s'$.

Dually the set $\mathbb{V}(\mathfrak{p}) \setminus (\cup_{\mathfrak{q} \in Q} \mathbb{V}(\mathfrak{q}))$ gets strictly larger in every step of AddPart and the algorithm works in such a way that $\mathbb{V}(\mathfrak{p}) \setminus (\cup_{\mathfrak{q} \in Q} \mathbb{V}(\mathfrak{q}))$ will always be a subset of S because in every step the set $\mathbb{V}(\mathfrak{p}) \setminus (\cup_{\mathfrak{q} \in Q} \mathbb{V}(\mathfrak{q}))$ is only enlarged with elements contained in some S_i. In particular $\mathbb{V}(\mathfrak{p}) \setminus (\mathbb{V}(\mathfrak{q}'_1) \cup \cdots \cup \mathbb{V}(\mathfrak{q}'_{s'})) \subset S$ or equivalently

$$(\overline{S} \setminus S) \cap \mathbb{V}(\mathfrak{p}) = \mathbb{V}(\mathfrak{p}) \setminus S \subset \mathbb{V}(\mathfrak{q}'_1) \cup \cdots \cup \mathbb{V}(\mathfrak{q}'_{s'}).$$

It remains to prove the inclusion "$\supset$" of Eq. (3). We have already observed above that $\mathbb{V}(\mathfrak{q}'_i) \subset \mathbb{V}(\mathfrak{p})$. So suppose for a contradiction that there exists $\mathfrak{q}' \in \{\mathfrak{q}'_1, \dots, \mathfrak{q}'_{s'}\}$ such that $\mathbb{V}(\mathfrak{q}')$ is not contained in $\overline{S} \setminus S$, or equivalently $\mathbb{V}(\mathfrak{q}') \cap S \neq \emptyset$. But then, as S is locally closed, $\mathbb{V}(\mathfrak{q}')$ is a non-empty open subset of $\mathbb{V}(\mathfrak{q}')$ and therefore

$$\mathbb{V}(\mathfrak{q}') = \overline{\mathbb{V}(\mathfrak{q}') \cap S} = \bigcup_{i=1}^{r} \overline{\mathbb{V}(\mathfrak{q}') \cap S_i}.$$

Please cite this article in press as: Montes, A., Wibmer, M., Gröbner bases for polynomial systems with parameters. Journal of Symbolic Computation (2010), doi:10.1016/j.jsc.2010.06.017

Table 6
AddPart algorithm.

$Q \leftarrow$ **AddPart**(H, C)
`Input:`
$H = \{\mathfrak{q}_1, \ldots, \mathfrak{q}_s\}$ set of prime ideals (the holes of some component $\mathfrak{p}$ of some S_i)
$C = \{C_j \;:\; 1 \le j \le l\}$ where $C_j = \big(\mathfrak{p}_j, \{\mathfrak{p}_{jk} \;:\; k = 1 \ldots r_j\}\big)$
are the P-representations which will be used to "fill the holes".
`Output:`
$Q = \{\mathfrak{q}'_1, \ldots, \mathfrak{q}'_{s'}\}$ the holes of the component $\mathfrak{p}$ of $S = S_1 \cup \cdots \cup S_r$

begin
$Q := H$
while there exists $\mathfrak{q} \in Q$ and $j \in \{1, \ldots, l\}$ with $\mathfrak{q} = \mathfrak{p}_j$ **do**

$$Q := \textsc{SelectMinIdeals}\left((Q \setminus \{\mathfrak{q}\}) \cup \{\mathfrak{p}_{j1}, \ldots, \mathfrak{p}_{jr_j}\}\right)$$

end while
while there exists $\mathfrak{q} \in Q$ and $j \in \{1, \ldots, l\}$
with $\mathfrak{q} \supset \mathfrak{p}_j$ and $\mathfrak{q} \not\supseteq \mathfrak{p}_{jk}$ for $k = 1, \ldots, r_j$ **do**

$$Q := \textsc{SelectMinIdeals}\left((Q \setminus \{\mathfrak{q}\}) \bigcup_{k=1}^{r_j} \textsc{PrimeDecomp}(\mathfrak{q} + \mathfrak{p}_{ik})\right)$$

end while
end

Since $\mathbb{V}(\mathfrak{q}')$ is irreducible we must have $\mathbb{V}(\mathfrak{q}') = \overline{\mathbb{V}(\mathfrak{q}') \cap S_i}$ for some $i \in \{1, \ldots, r\}$. As $\mathbb{V}(\mathfrak{q}') \subset \mathbb{V}(\mathfrak{q}_j)$ for some hole $\mathfrak{q}_j$ of $\mathfrak{p}$ with respect to S_{i_0} we have $\mathbb{V}(\mathfrak{q}') \cap S_{i_0} = \emptyset$ and therefore $i \neq i_0$. Furthermore since

$$S_i = \bigcup_{j=1}^{r^i} \left(\mathbb{V}(\mathfrak{p}^i_j) \setminus \mathbb{V}\big(\mathfrak{p}^i_{j1} \cap \cdots \cap \mathfrak{p}^i_{jr^i_j}\big)\right)$$

it follows

$$\mathbb{V}(\mathfrak{q}') = \bigcup_{j=1}^{r^i} \overline{(\mathbb{V}(\mathfrak{q}') \cap \mathbb{V}(\mathfrak{p}^i_j)) \setminus \mathbb{V}(\mathfrak{p}^i_{j1} \cap \cdots \cap \mathfrak{p}^i_{jr^i_j})}$$

and again by irreducibility of $\mathbb{V}(\mathfrak{q}')$ we obtain

$$\mathbb{V}(\mathfrak{q}') = \overline{(\mathbb{V}(\mathfrak{q}') \cap \mathbb{V}(\mathfrak{p}^i_j)) \setminus \mathbb{V}(\mathfrak{p}^i_{j1} \cap \cdots \cap \mathfrak{p}^i_{jr^i_j})} \tag{4}$$

for some $j \in \{1, \ldots, r^i\}$. In particular $\mathbb{V}(\mathfrak{q}') \subset \mathbb{V}(\mathfrak{q}') \cap \mathbb{V}(\mathfrak{p}^i_j)$ so that $\mathfrak{p}^i_j \subset \mathfrak{q}'$. Furthermore $\mathbb{V}(\mathfrak{q}') \not\subseteq \mathbb{V}(\mathfrak{p}^i_{jk})$ for $k = 1, \ldots, r^i_j$ because else the right-hand side of Eq. (4) would be the empty set. Summarily we have found i, j, $(i \neq i_0)$ such that $\mathfrak{q}' \supset \mathfrak{p}^i_j$ and $\mathfrak{q}' \not\supseteq \mathfrak{p}^i_{jk}$ for $k = 1, \ldots, r^i_j$. This contradicts our assumption that AddPart has terminated (Table 6). □

2.2. Representation of regular functions

In this section we will explain how we represent regular functions and how we can add and multiply them effectively. We also present two algorithms (Combine and Extend) which facilitate the conversion between different types of representations of regular functions.

Let $S \subset \overline{K}^m$ be a locally closed set and $f : S \to \overline{K}$ a regular function. By the very definition of a regular function (and quasi-compactness of locally closed sets) there exists a finite open covering $\{U_1, \ldots, U_r\}$ of S and polynomials $p_1, \ldots, p_r, q_1, \ldots, q_r \in K[\overline{a}]$ such that $f(a) = \frac{p_i(a)}{q_i(a)}$ for $a \in U_i$ and

Please cite this article in press as: Montes, A., Wibmer, M., Gröbner bases for polynomial systems with parameters. Journal of Symbolic Computation (2010), doi:10.1016/j.jsc.2010.06.017

ARTICLE IN PRESS

$i = 1, \dots, r$. Since we already know (see Section 2.1) how to represent the locally closed sets U_i we see that the data

$$\left\{\left(U_1, \frac{p_1}{q_1}\right), \dots, \left(U_r, \frac{p_r}{q_r}\right)\right\} \tag{5}$$

determines the regular function f. But this representation can be significantly improved. We can avoid to make the $U_i's$ explicit because the fractions $\frac{p_i}{q_i} \in K(\overline{a})$ can be chosen in such a way that they have the correct value on every point of S where they are defined (i.e. where the denominator q_i does not vanish).

Definition 21 (*Full Representation of Regular Functions*)**.** Let $f : S \to \overline{K}$ be a regular function on the locally closed set S. Let $p_1, \dots, p_r, q_1, \dots, q_r \in K[\overline{a}]$. We say that $(p_1, \dots, p_r; q_1, \dots, q_r)$ *is a full representation of* f if the following conditions are satisfied.

(i) $f(a) = \frac{p_i(a)}{q_i(a)}$ for every $a \in S$ with $q_i(a) \neq 0$,
(ii) for every $a \in S$ there exists $j \in \{1, \dots, r\}$ such that $q_j(a) \neq 0$ and
(iii) $p_i(a)q_j(a) = q_i(a)p_j(a)$ for all $a \in S$ and $1 \leq i, j \leq r$.

It follows from (ii) and (iii) that $p_i(a) = 0$ if $q_i(a) = 0$ for some $a \in S$. Note that it is not required that $S \setminus \mathbb{V}(q_i)$ is dense in S. We will see later in this section that every regular function admits a full representation as defined above. Conversely, it is obvious that if $p_1, \dots, p_r, q_1, \dots, q_r \in K[\overline{a}]$ satisfy conditions (ii) and (iii) then there exists a unique regular function $f : S \to \overline{K}$ such that $(p_1, \dots, p_r; q_1, \dots, q_r)$ is a full representation of f. In the examples we usually write the full representation more intuitively as $\{\frac{p_1}{q_1}, \dots, \frac{p_r}{q_r}\}$.

Definition 22 (*Full Representation of I-regular Functions*)**.** Let $S \subset \overline{K}^m$ be a locally closed set and $f : S \to \overline{K}[\overline{x}]$ an I-regular function. We say that a polynomial $\sum_\alpha c_\alpha \overline{x}^\alpha$ *is a full representation of* f if for every α the coefficient c_α is a full representation of $\mathrm{coef}(f, \alpha) \in \mathcal{O}(S)$.

In practice we will only have to deal with monic I-regular functions with $c_{\alpha_0} = 1$.

Definition 23 (*Generic Representation of Regular Functions*)**.** Let S be a locally closed set, $f : S \to \overline{K}$ a regular function and $p, q \in K[\overline{a}]$. We say that the pair $(p; q)$ is a *generic representation of* f if

(i) $S \setminus \mathbb{V}(q)$ is dense in S and
(ii) $f(a) = \frac{p(a)}{q(a)}$ for all $a \in S \setminus \mathbb{V}(q)$.

If $(p; q)$ is a generic representation of $f : S \to \overline{K}$ and $a \in S$ with $q(a) = 0$ then also $p(a) = 0$. To see this we observe that by the very definition of regular functions we can find polynomials $p', q' \in \overline{K}[\overline{x}]$ such that $q'(b) \neq 0$ and $f(b) = \frac{p'(b)}{q'(b)}$ for all b in an open neighborhood U of a in S. For $b \in U \cap (S \setminus \mathbb{V}(q))$ we have

$$\frac{p(b)}{q(b)} = f(b) = \frac{p'(b)}{q'(b)}$$

so that $(pq' - qp')(b) = 0$ for all $b \in U \cap (S \setminus \mathbb{V}(q))$. Since $S \setminus \mathbb{V}(q)$ is dense in S also $U \cap (S \setminus \mathbb{V}(q))$ is dense in U and so $(pq' - qp')(b) = 0$ for all $b \in U$. Since $a \in U$, $q(a) = 0$ and $q'(a) \neq 0$ we must have $p(a) = 0$.

Unfortunately it is not always possible to find a full representation $(p; q)$ of the regular function $f : S \to \overline{K}$ given by a single pair of polynomials (cf. Example 1). However, as we will see below, one can always find a generic representation of f. Also a generic representation $(p; q)$ of f already uniquely determines f. This is because if a regular function g which is defined on a dense open subset U of S can be extended to a larger open subset of S then this extension is unique. (Short proof: Let g_1, g_2 be extensions of g to an open subset V of S. Since U is dense in S the closure of U in V equals V. But U is contained in the closed subset $V' = \{a \in V : g_1(a) = g_2(a)\}$ of V so that $V = V'$, i.e. g_1 and g_2 agree on all of V.)

The advantage of the generic representation is that it is very convenient for computations, the disadvantage is that one cannot immediately determine the value of f at $a \in S$ if the denominator q vanishes at a.

For an I-regular function we can give a similar definition.

Please cite this article in press as: Montes, A., Wibmer, M., Gröbner bases for polynomial systems with parameters. Journal of Symbolic Computation (2010), doi:10.1016/j.jsc.2010.06.017

Table 7
Delta algorithm.

```
{δ_1, ..., δ_s, δ} ← Delta(p_1, ..., p_s)
Input:
    p_1, ..., p_s ⊂ K[ā] prime ideals
    It is assumed that p_1 ∩ ··· ∩ p_s is a minimal prime decomposition.
Output:
    {δ_1, ..., δ_s, δ} ⊂ K[ā] such that δ_i(a) = 0 on ⋃_{j≠i} V(p_j),
      δ(a) = δ_i(a) ≠ 0 on U_i ⊂ V(p_i) with closure(U_i) = V(p_i)

begin
  a_1 := p_1;  b_s := p_s
  for i = 2 ... s − 1 do a_i := a_{i−1} ∩ p_i
  for i = s − 1 ... 2 do b_i := p_i ∩ b_{i+1}
  h_1 := b_2;  h_s := a_{s−1}
  for i = 2 ... s − 1 do h_i := a_{i−1} ∩ b_{i+1}
  for i = 1 ... s choose δ_i an element of gb(h_i) that does not lie in p_i
  δ := Σ_{i=1}^{s} δ_i
end
```

Definition 24 (*Generic Representation of I-regular Functions*)**.** Let $F : S \to \overline{K}[\overline{x}]$ be a monic I-regular function on the locally closed set S. We say that $P \in K[\overline{a}][\overline{x}]$ is a *generic representation* of F if

(i) $S \setminus \mathbb{V}(q)$ is dense in S, where $q = \mathrm{lc}(P) \in K[\overline{a}]$
(ii) $F(a, \overline{x}) = \frac{P(a,\overline{x})}{q(a)}$ for all $a \in S \setminus \mathbb{V}(q)$.
(iii) $P(a, \overline{x}) = 0$ for all $a \in \mathbb{V}(q) \cap S$.

The purpose of algorithm Combine is to compute a generic representation. And the task of algorithm Extend is to compute a full representation from a generic representation.

Computing a generic representation of a regular function $f : S \to \overline{K}$ is a special case of the computation of a generic representation of a monic I-regular function $F : S \to \overline{K}[\overline{x}]$. So the algorithm Combine is designed for the second option, and is nothing else than a Chinese remainder method (Becker and Weispfenning, 1991).

Before using Combine, a previous algorithm Delta must be applied.

Lemma 25 (*Delta*)**.** *Let $\{\mathfrak{p}_1, \ldots, \mathfrak{p}_s\}$ be a minimal prime decomposition. Then the algorithm* Delta *of Table 7 computes polynomials $\{\delta_1, \ldots, \delta_s, \delta\} \subset K[\overline{a}]$ such that*

(i) *$\delta_i(a) \neq 0$ for all a in an open subset $a \in U_i \subset \mathbb{V}(\mathfrak{p}_i)$, i.e. $\overline{U_i} = \mathbb{V}(\mathfrak{p}_i)$,*
(ii) *$\delta_i(a) = 0$ for all $a \in (\mathbb{V}(\mathfrak{p}_i) \setminus U_i) \bigcup \left(\bigcup_{j\neq i} \mathbb{V}(\mathfrak{p}_j)\right)$,*
(iii) *$\delta(a) \neq 0$ for all a in an open and dense subset $a \in U = \bigcup_j U_j \subset \bigcup_j \mathbb{V}(\mathfrak{p}_j)$, and $\delta(a) = \delta_i(a)$ for all $a \in U_i$,*
(iv) *$\delta(a) = 0$ for all $a \in \left(\bigcup_j \mathbb{V}(\mathfrak{p}_j)\right) \setminus U$.*

Proof. The algorithm computes $\mathfrak{h}_i = \bigcap_{j\neq i} \mathfrak{p}_j$. Thus if $h \in \mathfrak{h}_i$ then $h(a) = 0$ for all $a \in \bigcup_{j\neq i} \mathbb{V}(\mathfrak{p}_j)$. Then it chooses a δ_i of $\mathrm{gb}(\mathfrak{h}_i)$ that does not lie in $\mathfrak{p}_i$, so that we have $\delta_i(a) \neq 0$ on an open subset $U_i \subset \mathbb{V}(\mathfrak{p}_i)$, and $\delta_i(a) = 0$ for all $a \in (\mathbb{V}(\mathfrak{p}_i) \setminus U_i) \bigcup \left(\bigcup_{j\neq i} \mathbb{V}(\mathfrak{p}_j)\right)$. Finally δ is the sum of all the δ_i and thus it has the desired properties. □

Now we are prepared to present algorithm Combine (see Table 8), whose action is summarized in the following

Lemma 26 (*Combine*)**.** *Let $F : S \to K[\overline{x}]$ be a monic I-regular function on the locally closed segment S whose components are $\{\mathfrak{p}_1, \ldots, \mathfrak{p}_s\}$. Let $\{\delta_1, \ldots, \delta_r, \delta\} \subset K[\overline{a}]$ be the output functions of* Delta *applied to S, and assume that we are given polynomials $P_i \in K[\overline{a}][\overline{x}]$, $i = 1 \ldots s$ such that*

$$\mathrm{lt}(P_i) = q_i(\overline{a})x^{\alpha_0},$$
$$\text{where } q_i = \mathrm{lc}(P_i), \quad x^{\alpha_0} = \mathrm{lpp}(P_i) = \mathrm{lpp}(F),$$
$$P_i(a, \overline{x})/q_i(a) = F(a, \overline{x}) \quad \text{for all } a \in \mathbb{V}(\mathfrak{p}_i) \cap S.$$

Table 8
Combine algorithm.

```
P ← Combine ((p_1, P_1, δ_1), ..., (p_s, P_s, δ_s), δ)
Input:
    p_1, ..., p_s ⊂ K[ā] are the components of the locally closed segment S, (i.e. S̄ = V(⋂_i p_i)),
    P_1, ..., P_s ∈ K[ā][x̄] with lt(P_i) = q_i x^{α_0} and q_i(a) ≠ 0 on a non-empty open subset of V(p_i)
        where P_i(a, x̄)/q_i(a) = F(a, x̄), and F is a monic I-regular function F : S → K̄[x̄], δ_1, ..., δ_s, δ ∈ K[ā]
        are the output of the algorithm Delta.
Output: P ∈ K[ā][x̄] a generic representation of F on S

begin
  For i ∈ {1, ..., s} set
    q_i := lc(P_i)
    J_i := {k ∈ {1, ..., s} : q_i ∈ p_k}
    q̃_i = q_i + Σ_{j∈J_i} δ_j
  P̃ := (P_1 δ_1)/(q̃_1 δ) + ··· + (P_s δ_s)/(q̃_s δ)
  P = eliminate denominators(P̃)
end
```

Then the algorithm Combine *on Table 8 computes a generic representation* $P \in K[\overline{a}][\overline{x}]$ *of* F *on* S *with* $\mathrm{lc}(P) = q$ *so that* $P(a, \overline{x})/q(a) = F(a, \overline{x})$ *on each point* a *of an open and dense subset of* S.

Proof. Let U_i and $U = \bigcup_j U_j$ be the segments where the $\delta_i(a)$ have the desired properties. Taking into account the properties of $\delta_i(a)$, the polynomial $\tilde{q}_i$ verifies:

$$\begin{array}{ll} \tilde{q}_i(a) = q_i(a) & \text{if } a \in \mathbb{V}(\mathfrak{p}_i) \\ \tilde{q}_i(a) = q_i(a) & \text{if } a \in \mathbb{V}(\mathfrak{p}_k) \text{ and } q_i \notin \mathfrak{p}_k \\ \tilde{q}_i(a) = \delta_i(a) & \text{if } a \in \mathbb{V}(\mathfrak{p}_k) \text{ and } q_i \in \mathfrak{p}_k. \end{array}$$

Thus $\frac{P_i(a,\overline{x})\delta_i(a)}{\tilde{q}_i(a)\delta(a)}$ has a denominator that is non-null for all $a \in U' = S \cap U \subset S$, where U' is an open and dense subset of S and is null on $\overline{S} \setminus U'$. For $a \in U'_i = U_i \cap S$ there is $\tilde{q}_i(a) = q_i(a)$ and $\delta_i(a) = \delta(a)$ and thus

$$\frac{P_i(a, \overline{x})\delta_i(a)}{\tilde{q}_i(a)\delta(a)} = \frac{P_i(a, x)}{q_i(a)}$$

and for $a \in U'_k$ for $k \neq i$ is $\delta_i(a) = 0$ and the denominator is non-zero, so that

$$\frac{P_i(a, \overline{x})\delta_i(a)}{\tilde{q}_i(a)\delta(a)} = 0.$$

Thus adding together all these terms and eliminating denominators, the polynomial will be non-zero on U' and 0 on $\overline{S} \setminus U'$ and the result follows. □

Example 27. Let $\mathfrak{a} = \mathfrak{p}_1 \cap \mathfrak{p}_2 \subset K[a_1, a_2]$ with $\mathfrak{p}_1 = \langle a_1 \rangle$, $\mathfrak{p}_2 = \langle a_2 \rangle$ and

$$S = \mathbb{V}(a_1 a_2) \setminus (\mathbb{V}(a_1, a_2 - 1) \cup \mathbb{V}(a_1 - 4, a_2) \cup \mathbb{V}(a_1, a_2)).$$

Define a monic I-regular function $F : S \to \overline{K}[x]$ by $P_1 = (a_2 - 1)x + (a_2^2 - 4)$ on $\mathbb{V}(a_1) \setminus (\mathbb{V}(a_1, a_2 - 1) \cup \mathbb{V}(a_1, a_2))$ and $P_2 = (a_1 - 4)x + (a_1^3 - 16)$ on $\mathbb{V}(a_2) \setminus (\mathbb{V}(a_1 - 4, a_2) \cup \mathbb{V}(a_1, a_2))$. We compute first Delta and obtain $\delta_1 = a_2$, $\delta_2 = a_1$, $\delta = a_1 + a_2$. Then we apply Combine and obtain:

$$\begin{aligned} \tilde{P} &= \frac{a_2}{a_1 + a_2} \frac{(a_2 - 1)x + (a_2^2 - 4)}{a_2 - 1} + \frac{a_1}{a_1 + a_2} \frac{(a_1 - 4)x + (a_1^3 - 16)}{a_1 - 4} \\ &= \frac{(a_1 a_2^2 - 5a_1 a_2 - 4a_2^2 + 4a_2 + a_1^2 a_2 + a_1^2 + 4a_1)x + a_1^4 a_2 - a_1^4 + a_1 a_2^3 - 20a_1 a_2 + 16a_1 - 4a_2^3 + 16a_2}{(a_1 + a_2)(a_1 - 4)(a_2 - 1)}. \end{aligned}$$

Eliminating denominators and reducing modulo $\mathfrak{a}$ we obtain

$$P = (-a_1^2 + 4a_1 - 4a_2^2 + 4a_2)x + (-a_1^4 + 16a_1 - 4a_2^3 + 16a_2).$$

Table 9
Extend algorithm.

$(p_1, \ldots, p_s; q_1, \ldots, q_s) \leftarrow$ **Extend**(S, p, q)
`Input:`
$S \subset \overline{K}^m$ locally closed
$p, q \in K[\overline{a}]$ such that

- $S \setminus \mathbb{V}(q)$ is dense in S,
- there exists a regular function $f : S \to \overline{K}$ such that $f(a) = \frac{p(a)}{q(a)}$ for every $a \in S \setminus \mathbb{V}(q)$.

`Output:` A full representation of f

begin
Let $\mathfrak{a} \subset K[\overline{a}]$ be the radical ideal such that $\mathbb{V}(\mathfrak{a}) = \overline{S}$, and $\begin{pmatrix} p_1 \\ q_1 \end{pmatrix}, \ldots, \begin{pmatrix} p_s \\ q_s \end{pmatrix}$ a generating set of the $K[\overline{a}]$-module

$$\left\{ \begin{pmatrix} g \\ h \end{pmatrix} \in K[\overline{a}]^2 : gq + h(-p) \in \mathfrak{a} \right\}$$

which describes the syzygies of $(q, -p)$ modulo $\mathfrak{a}$.
end

We observe that P specializes to non-null in all $\mathbb{V}(\mathfrak{a})$ except the points (0, 0), (0, 1), (4, 0), and when normalized, specializes to the normalized P_1 on $\mathbb{V}(a_1)$ and to the normalized P_2 on $\mathbb{V}(a_2)$, and is 0 on the excluded points (0, 0), (0, 1), (4, 0).

If the generic representation $(p; q)$ of $f : S \to \overline{K}^m$ obtained by Combine algorithm does not satisfy $q(a) \neq 0$ for all $a \in S$, then we can use the following algorithm Extend to compute a complete representation $(p_1, \ldots, p_s; q_1, \ldots, q_s)$ of f.

We note that the $K[\overline{a}]$-module defined in algorithm Extend on Table 9 can be computed using standard Gröbner bases techniques.

Proposition 28 (*Extend Algorithm*)**.** *Let $S \subset \overline{K}^m$ be locally closed and $f : S \to \overline{K}$ a regular function. Let $p, q \in K[\overline{a}]$ such that $S \setminus \mathbb{V}(q)$ is dense in S and $f(a) = \frac{p(a)}{q(a)}$ for all $a \in S \setminus \mathbb{V}(q)$. Then algorithm* Extend *computes a full representation of f.*

Proof. We have to show that $(p_1, \ldots, p_s; q_1, \ldots, q_s) = \text{Extend}(S, p, q)$ is a representation of f. Let $i \in \{1, \ldots, s\}$. By construction $p_i q - q_i p \in \mathfrak{a}$ so that $\frac{p_i(a)}{q_i(a)} = \frac{p(a)}{q(a)}$ for all $a \in S \setminus \mathbb{V}(qq_i)$. Since $S \setminus \mathbb{V}(q)$ is dense in S we see that $(S \setminus \mathbb{V}(q)) \cap (S \setminus \mathbb{V}(q_i)) = S \setminus \mathbb{V}(qq_i)$ is dense in $S \setminus \mathbb{V}(q_i)$. Therefore, if the regular function defined by $\frac{p}{q}$ on $S \setminus \mathbb{V}(qq_i)$ can be extended to $S \setminus \mathbb{V}(q_i)$ then this extension is unique. But both f and $\frac{p_i}{q_i}$ define such extensions. Consequently $f(a) = \frac{p_i(a)}{q_i(a)}$ for all $a \in S \setminus \mathbb{V}(q_i)$ and (i) of Definition 21 is proved.

To verify (ii) fix $a \in S$. The open subsets of S of the form $S \setminus \mathbb{V}(q')$ with $q' \in K[\overline{a}]$ are a basis of the topology of S. Thus there exists polynomials $q', \widetilde{p}, \widetilde{q} \in K[\overline{a}]$ such that $a \in S \setminus \mathbb{V}(q')$ and $f(b) = \frac{\widetilde{p}(b)}{\widetilde{q}(b)}$ for all $b \in S \setminus \mathbb{V}(q')$. In particular $\widetilde{q}(b) \neq 0$ for all $b \in S \setminus \mathbb{V}(q')$. This means that $\mathbb{V}(\widetilde{q}) \cap S \subset \mathbb{V}(q') \cap S$. If $\mathfrak{a} \subset K[\overline{a}]$ is the radical ideal with $\mathbb{V}(\mathfrak{a}) = \overline{S}$ then $\mathbb{V}(\mathfrak{a} + \langle \widetilde{q} \rangle) \subset \mathbb{V}(\mathfrak{a} + \langle q' \rangle)$. Therefore $q' \in \sqrt{\mathfrak{a} + \langle \widetilde{q} \rangle}$. So we can find $n \geq 1$, $h \in K[\overline{a}]$ and $g \in \mathfrak{a}$ such that $q'^n = h\widetilde{q} + g$. Then we have for $b \in S \setminus \mathbb{V}(q')$

$$f(b) = \frac{\widetilde{p}(b)}{\widetilde{q}(b)} = \frac{\widetilde{p}(b)h(b)}{\widetilde{q}(b)h(b)} = \frac{p'(b)}{q'(b)^n}$$

with $p' = \widetilde{p}h$.

We claim that $(p'q - q'^n p)q'$ lies in $\mathfrak{a}$. Because $S \setminus \mathbb{V}(q)$ is dense in $\mathbb{V}(\mathfrak{a})$ it suffices to see that $(p'(b)q(b) - q'^n(b)p(b))q'(b) = 0$ for all $b \in S \setminus \mathbb{V}(q)$. But if $b \in S \cap \mathbb{V}(q')$ this is trivial and if $b \in S \setminus \mathbb{V}(q'q)$ then $p'(b)q(b) - q'(b)^n p(b) = 0$ because

$$\frac{p'(b)}{q'(b)^n} = f(b) = \frac{p(b)}{q(b)}.$$

Consequently there exist $h_1, \ldots, h_s \in K[\overline{a}]$ such that

$$\begin{pmatrix} p'q' \\ q'^{n+1} \end{pmatrix} = h_1 \begin{pmatrix} p_1 \\ q_1 \end{pmatrix} + \cdots + h_s \begin{pmatrix} p_s \\ q_s \end{pmatrix}.$$

Now if $q_i(a)$ was equal to zero for every $i \in \{1, \ldots, r\}$ then also $q'(a)$ would be equal to zero which is not the case as $a \in S \setminus \mathbb{V}(q')$. Therefore (ii) is proved.

Finally to verify (iii) of Definition 21 let $i, j \in \{1, \ldots, s\}$. Multiplying the equation $p_i q - q_i p \in \mathfrak{a}$ with q_j and replacing $q_j p$ with $p_j q$ we obtain $q_j p_i q - q_i p_j q \in \mathfrak{a}$ so that

$$q(p_i q_j - q_i p_j) \in \mathfrak{a} = \mathfrak{p}_1 \cap \cdots \cap \mathfrak{p}_r.$$

Since $q \notin \mathfrak{p}_1, \ldots, \mathfrak{p}_r$ we see that $p_i q_j - q_i p_j \in \mathfrak{a}$ and (iii) is proved. □

Example 29. To understand the power of Extend algorithm, we apply it to the result of Example 27, even if it was not necessary. Applying Extend to the generic representation obtained for $f : S \to \overline{K}$ we obtain the pair of syzygies:

$$\left(a_2^3 - 4a_2, a_1^3 + 4a_2^2 - 16; a_2^2 - a_2, a_1 + 4a_2 - 4\right).$$

Observe that whether the first pair is zero on $\mathbb{V}(\mathfrak{p}_1)$, the second one is non-null in all points of S and it extends f to $S' = S \cup \{(0, 0)\}$, as it assigns to $f(0, 0)$ the value 4, as expected by the initial data. So we only need the second syzygy.

$$\frac{p}{q} = \frac{a_1^3 + 4a_2^2 - 16}{a_1 + 4a_2 - 4}.$$

This provides the full representation of the I-regular function

$$P = (a_1 + 4a_2 - 4)x + (a_1^3 + 4a_2^2 - 16)$$

showing that f can be defined as regular function in the larger set $S' \supset S$.

2.3. Computations with I-regular functions

For performing computations with I-regular functions the generic representation is very practical as the addition and multiplication of the regular functions can be performed by the usual computations in $K(\overline{a})$.

If P is a generic representation of $f : S \to \overline{K}[\overline{x}]$ and $a \in S$ with $\mathrm{lc}(P)(a) = 0$ then $P(a, \overline{x}) = 0$. This follows immediately from Definition 24.

Using generic representations it is very easy to perform computations like reduction with monic I-regular functions. The disadvantage of the generic representation is that the value of f at $a \in S$ cannot immediately by determined from p if $\mathrm{lc}(p)(a) = 0$. However we only need to apply Extend to the coefficients to convert a generic representation into a full representation. I.e.

$$\sum_{\alpha} \textsc{Extend}(S, p_\alpha, \mathrm{lc}(p))\overline{x}^\alpha$$

is a full representation of f.

Computations (like reduction) with monic I-regular functions are most easily performed using the generic representation. The actual computations take place in $K[\overline{a}, \overline{x}]$ and the operations with the I-regular functions simply correspond to the usual operations in $K[\overline{a}, \overline{x}]$ only that we occasionally have to multiply with the leading coefficient of a generic representation to avoid denominators.

3. The GCover algorithm

In this section we describe the algorithm GCover (see Table 10). It is the heart of the main algorithm GröbnerCover. Algorithm GCover takes as input a finite set of homogeneous polynomials

Please cite this article in press as: Montes, A., Wibmer, M., Gröbner bases for polynomial systems with parameters. Journal of Symbolic Computation (2010), doi:10.1016/j.jsc.2010.06.017

Table 10
GCover algorithm.

```
G = ((S_i, B_i) : i = 1, ..., s) ← GCover(F, ≻_x̄).
Input: F ⊂ K[ā][x̄]: finite set of homogeneous polynomials generating the ideal I
Output: G = ((S_i, B_i) : i = 1, ..., s): the canonical Gröbner cover of I,
            the basis elements are given in generic representation

begin
  BT := the terminal vertices of BuildTree(F)
  L := the list of all lpp's occurring in BT
  G := ∅
  for each T ∈ L do
    M := the list of all segments ((𝔞, h), B) given in BT with lpp equal T
    H := M without the bases
    S := LCUnion(H)
    Let 𝔭_1, ..., 𝔭_r denote the components of S.
    For i = 1, ..., r let ((𝔞_i, h_i), B_i) denote the unique segment in M such that 𝔞_i has component 𝔭_i.
      #{This has already been computed internally by LCUnion}
    B := Basis(((𝔭_1, B_1), ..., (𝔭_r, B_r)))
    G := G ∪ {(S, B)}
  end do
end
```

(with respect to the variables) which generate the ideal I and computes the canonical Gröbner cover of I as given in Theorem 8. The I-regular functions in the bases are given in generic representation.

Throughout Section 3 we assume that $I \subset K[\overline{a}][\overline{x}]$ is a homogeneous ideal given by a finite set of homogeneous generators. The case of non-homogeneous ideals will be treated in Section 4. Of course we also have a fixed term-order $\succ_{\overline{x}}$ on the variables but it will usually not be indicated.

We now describe the action of GCover. It has three main steps: BuildTree, LCUnion and Basis. We start with the call to BuildTree. Algorithm BuildTree (described in Theorem 30 in Section 3.2) is the first part of our main algorithm. It builds a discussion tree, whose terminal vertices contain a disjoint, reduced comprehensive Gröbner system. This reduced Gröbner system can equivalently be interpreted as a Gröbner cover where the elements of the bases are given in a generic representation. The lpp-segments are partitioned into smaller segments and in each of these segments the I-regular functions in the Gröbner basis can be fully represented by a single polynomial. More explicitly the result of BuildTree consists of a set of pairs $\{(S_i, B_i) : 1 \leq i \leq s\}$, where the S_i are parametric subsets of $\overline{K}^m$ given in R-representation (see Section 2.1), and the B_i are subsets of $K[\overline{a}][\overline{x}]$ such that, $\text{lpp}(B_i)$ is the minimal generating set of $\text{lpp}(S_i)$ and evaluating the elements of B_i at $a \in S_i$ yields the reduced Gröbner basis of the specialized ideal I_a up to normalization.

In the next step we group together all the segments S_i with the same leading power products: From Theorem 7 we know that the union of all S_i's with the same lpp is locally closed and parametric. Thus we can use algorithm LCUnion to compute this union. First we transform the R-representations of the S_i's into P-representations and then we apply algorithm LCUnion to obtain the complete lpp-segments in P-representation. Thus we have already found the segments of the canonical Gröbner cover.

It remains to compute the generic representations of the basis elements. This is the task of algorithm Basis which is described in Section 3.3.

3.1. Auxiliary algorithms

In this subsection we discuss two algorithms that are used throughout the whole computations: Pdiv and PNormalForm.

Pdiv is the Hironaka reduction of a polynomial $p \in K[\overline{a}][\overline{x}]$ modulo $\{p_1, \ldots, p_s\}$ over a locally closed segment $S = \mathbb{V}(\mathfrak{a}) \setminus \mathbb{V}(\mathfrak{b})$. For details see Montes (2002). It is assumed that, for all $a \in S$ and for all i, $\text{lc}(p_i)(a) \neq 0$. The reduction is:

$$hp = q_1 p_1 + \cdots + q_s p_s + r \tag{6}$$

Please cite this article in press as: Montes, A., Wibmer, M., Gröbner bases for polynomial systems with parameters. Journal of Symbolic Computation (2010), doi:10.1016/j.jsc.2010.06.017

Table 11
BuildTree algorithm.

$T \leftarrow$ **BuildTree**(F)
Input: $F \subset K[\overline{a}][\overline{x}]$ a finite set of homogeneous polynomials generating the ideal I
Output: T: the tree where all data are stored. At the terminal vertices these data form a disjoint Gröbner cover.

begin
  $T :=$ a global empty tree
  $((\mathfrak{a}, h), B, l, P) := ((\langle 0\rangle, 1), F, 0, \emptyset)$
  Let r be the root node of the initially empty tree T.
  RecBuildTree$(r, (\mathfrak{a}, h), B, l, P)$
end

satisfying

1. $q_1, \ldots, q_s, r \in K[\overline{a}][\overline{x}]$,
2. $h \in K[\overline{a}]$ is a power product in $\text{lc}(p_1), \ldots, \text{lc}(p_s)$,
3. $\text{lpp}(q_i p_i) \le \text{lpp}(p)$ for $i = 1, \ldots, s$,
4. No power product in the support of r is divisible by $\text{lpp}(p_i)$ for $i = 1, \ldots, s$.

It is easy to prove (Montes, 2002) that the specialization of the Pdiv reduction for any $a \in S$

$$h(a)p(a, \overline{x}) = q_1(a, \overline{x})p_1(a, \overline{x}) + \cdots + q_s(a, \overline{x})p_s(a, \overline{x}) + r(a, \overline{x})$$

is the usual division of $p(a, \overline{x})$ by $\{p_1(a, \overline{x}), \ldots, p_s(a, \overline{x})\}$ on $\overline{K}[\overline{x}]$. The input–output scheme of algorithm Pdiv is

$$r \leftarrow \textbf{Pdiv}(p, \{p_1, \ldots, p_s\}).$$

Given a polynomial $p \in K[\overline{a}][\overline{x}]$ and the R-representation $(\mathfrak{a}, h)$ of a locally closed subset S the second algorithm PNormalForm computes a "normalform" $r \in K[\overline{a}, \overline{x}]$ of p on S. It first reduces the coefficients of p modulo $\mathfrak{a}$ and then eliminates all factors of p that are elements of $K[\overline{a}]$ and are non-null on all points of S.

The input–output scheme is

$$r \leftarrow \textbf{PNormalForm}(p, (\mathfrak{a}, h)).$$

3.2. *The* BuildTree *algorithm*

We begin now the discussion of the first crucial part of our algorithm GCover, namely the algorithm BuildTree (see Table 11).

This subsection is organized in descending design. So we present first the main algorithm BuildTree, then the recursive algorithm RecBuildTree called by BuildTree, and finally the two sub-algorithms DiscussPolys and DiscussSPolys used by RecBuildTree. At the end we also detail the auxiliary algorithms ReduceGB. It is recommended to read this section first in the given order (without regarding the proofs) and then read the proofs in the opposite order: DiscussPolys, DiscussSpolys and finally BuildTree.

BuildTree is a Buchberger like algorithm for computing a Gröbner basis. As here the coefficients of the polynomials are polynomials in the parameters, the algorithm branches every time when it has to deal with a polynomial of the basis or an S-polynomial whose leading coefficient vanishes at some, but not at all points of the locally closed set under consideration. It builds up a dichotomic binary tree, whose branches at each vertex correspond to the annihilation or not of a new polynomial of $K[\overline{a}]$. So, at a vertex, some polynomials, say $N \subset K[\overline{a}]$ have been assumed to be null and some others, say $W \subset K[\overline{a}]$, have been assumed to be non-null. This determines a locally closed subset S of $\overline{K}^m$, of the special kind for which R-representations can be used (see Section 2.1). i.e.

$$S = \mathbb{V}(N) \setminus \mathbb{V}(h) \subset \overline{K}^m, \quad \text{with } h = \prod_{w \in W} w \in K[\overline{a}].$$

A vertex of the tree is given by a list of zeros and ones which describes its position in the tree. At each vertex of the tree BuildTree stores the *vertex data* $((\mathfrak{a}, h), B, l, P)$. Where

Please cite this article in press as: Montes, A., Wibmer, M., Gröbner bases for polynomial systems with parameters. Journal of Symbolic Computation (2010), doi:10.1016/j.jsc.2010.06.017

Table 12
RecBuildTree algorithm.

```
RecBuildTree(v, (𝔞, h), B, l, P)
Input:
    v: current vertex of the global tree T at which RecBuildTree is called ((𝔞, h), B, l, P): the vertex data of v
Output:
    Builds recursively the tree T, storing the vertex data at the vertices.

begin
    Store ((𝔞, h), B, l, P) in v.
    B' := B
    if l < |B| then
        (B', (𝔞0, h0), l0, P0, (𝔞1, h1), l1, P1) := DiscussPolys((𝔞, h), B, l, P)
        l := l0
    end if
    if l = |B'| and B' ≠ ∅ then
        (B', (𝔞0, h0), l0, P0, (𝔞1, h1), l1, P1) := DiscussSPolys(B', (𝔞, h), l, P)
    end if
    if 𝔞0 ≠ ⟨1⟩ then
        Create two new vertices v0 and v1 descending from v
        RecBuildTree(v0, (𝔞0, h0), B', l0, P0)
        RecBuildTree(v1, (𝔞1, h1), B', l1, P1)
    else    # {then P = ∅}
        B' := RedGB(MinGB(B'))
        Store ((𝔞, h), B') in v.
    end if
end
```

- $(\mathfrak{a}, h)$ is an R-representation of $S = V(\mathfrak{a}) \setminus \mathbb{V}(h)$,
- B is a finite list of polynomials in $K[\overline{a}, \overline{x}]$ such that for every $a \in S$ the polynomials obtained from B by specialization are a generating set of I_a. The i-th element in this list will be denoted with $B[i]$.
- $0 \leq l \leq |B|$ is an integer such that for $i = 1, \ldots, l$ we have $\mathrm{lc}(B[i])(a) \neq 0$ for all $a \in S$ and so far the algorithm has not obtained information about the vanishing behavior of $\mathrm{lc}(B[l+1])$ on S,
- P is a list of pairs of elements of $\{1, \ldots, l\}$ such that for each pair $(i, j) \in P$ the S-polynomial of $B[i]$ and $B[j]$ has not yet been considered in the algorithm.

Using R-representations it is very easy to split recursively into two dichotomic branches when the algorithm has to decide if a new polynomial $f \in K[\overline{a}]$ is null or non-null on the given locally closed set S. This is done by the recursive algorithm RecBuildTree that uses the algorithm Split (see Table 3) already discussed in Section 2.

Theorem 30 (*BuildTree Algorithm*)**.** *Given a finite set $F \subset K[\overline{a}][\overline{x}]$ of homogeneous polynomials generating the ideal I, the algorithm* BuildTree *builds a finite binary tree T with root such that at each terminal vertex v of T the data $((\mathfrak{a}_v, h_v), B_v)$ with the following properties is stored.*

(i) *$(\mathfrak{a}_v, h_v)$ is an R-representation of the locally closed set $S_v = S((\mathfrak{a}_v, h_v))$ and B_v is a finite subset of $K[\overline{a}, \overline{x}]$.*
(ii) *S_v is parametric,* $\mathrm{lpp}(B_v)$ *is the minimal generating set of* $\mathrm{lpp}(S_v)$ *and B_v specializes to the reduced Gröbner basis of I_a (up to normalization) for every $a \in S_v$.*
(iii) *The S_v's are pairwise disjoint and cover the whole $\overline{K}^m$ (as v ranges over all terminal vertices).*

So in essence the terminal vertices of BuildTree *give a disjoint Gröbner cover of $\overline{K}^m$ with respect to I.*

Proof. The algorithm BuildTree only creates the root vertex of the tree T and then calls the recursive algorithm RecBuildTree (see Table 12).

If RecBuildTree is called at vertex v with the vertex data $((\mathfrak{a}, h), B, l, P)$ then either v becomes a terminal vertex or the algorithm has to split, so that v has two successor vertices v_0 and v_1 and RecBuildTree calls itself at v_0, v_1 with the new vertex data $((\mathfrak{a}_0, h_0), B_0, l_0, P_0)$, $((\mathfrak{a}_1, h_1), B_1, l_1, P_1)$ respectively. We note that in the second case we have $\mathfrak{a}_0 \neq \langle 1 \rangle$ and $\mathfrak{a}_0 \supsetneqq \mathfrak{a}$ by Lemmas 31 and 32.

If we follow RecBuildTree along a path from the root down the tree then we find that it essentially performs the usual Buchberger algorithm: First RecBuildTree repeatedly calls DiscussPolys until we obtain a basis B such that the leading coefficient of every polynomial in B is non-zero at every point of the current locally closed set S (i.e. $l = |B|$). Then we call DiscussSPolys and advance in the Buchberger algorithm. If at some vertex v we take the right branch to v_0 when DiscussSPolys has found a splitting, then the new vertex data $(\mathfrak{a}_0, h_0), B_0, l_0, P_0)$ at v_0 satisfies $l_0 = |B_0|-1$ and DiscussPolys will be called. If we take the right branch to v_1 then the new vertex data $((\mathfrak{a}_1, h_1), B_1, l_1, P_1)$ satisfies $l_1 = |B_1|$ and DiscussSPolys will advance in the Buchberger algorithm.

We now prove that the tree is finite, i.e. the algorithm terminates. Suppose that BuildTree creates an infinite tree. Then there is an infinite path starting from the root. At a vertex v of the path, the path can either turn left to v_1 or right to v_0. If the path would turn right an infinite number of times then we would obtain an infinite strictly increasing sequence of ideals in $K[\overline{a}]$ which is not possible. Thus from a certain vertex onwards the path always keeps left, making only new non-null assumptions. But then the finiteness follows from the termination of the usual Buchberger algorithm.

Suppose the algorithm eventually reaches a terminal vertex v. This can only happen if the current list P is empty and we see from Lemma 32 that for each a in the current locally closed subset $S = S_v$ the current basis B' specializes to a Gröbner basis of I_a. Now for RecBuildTree it only remains to minimize and to reduce the basis. The algorithm stores the new basis in v and quits. Thus, as claimed in (ii), we see that $\mathrm{lpp}(B_v)$ is the minimal generating set of $\mathrm{lpp}(I_a)$ and that B_v specializes, up to normalization, to the reduced Gröbner basis of I_a for every $a \in S_v$. Next we will prove that S_v is parametric. In general the elements of B_v need not lie in I, because we have reduced them modulo the assumed null-conditions. But by construction for every $p \in B_v$ there are polynomials $p' \in I$ and $q' \in K[\overline{a}]$ such that $q'(a) \neq 0$ and $q'(a)p(a, \overline{x}) = p'(a, \overline{x})$ for all $a \in S_v$ and so

$$\frac{p(a,\overline{x})}{\mathrm{lc}(p)(a)} = \frac{q'(a)p(a,\overline{x})}{q'(a)\mathrm{lc}(p)(a)} = \frac{p'(a,\overline{x})}{\mathrm{coef}(p',\mathrm{lpp}(p))(a)}$$

for all $a \in S_v$ and we see that S_v is parametric (cf. Remark 5).

Claim (i) is obvious and Claim (iii) is immediate from the algorithm and Lemmas 31 and 32. □

If we are given the vertex data $((\mathfrak{a}, h), B, l, P)$ then we already know that, for all $a \in S((\mathfrak{a}, h))$ and $i = 1, \dots, l$, is $\mathrm{lc}(B[i])(a) \neq 0$. The task of DiscussPolys (see Table 13) is to obtain new information about the vanishing behavior of the leading coefficients of the next polynomials $B[l+1], B[l+2], \dots$ in the list until a splitting is necessary. The result of DiscussPolys is summarized in the following

Lemma 31 (*DiscussPolys Algorithm*)**.** *Suppose that* DiscussPolys *is called with the vertex data* $((\mathfrak{a}, h), B, l, P)$*. Then two new vertex data* $((\mathfrak{a}_0, h_0), B', l_0, P_0)$ *and* $((\mathfrak{a}_1, h_1), B', l_1, P_1)$ *with the following properties are obtained:*

(i) $S = S_0 \uplus S_1$ *where* $S = S((\mathfrak{a}, h))$, $S_0 = S((\mathfrak{a}_0, h_0))$ *and* $S_1 = S((\mathfrak{a}_1, h_1))$,
(ii) – *either* $\mathfrak{a}_0 = \langle 1\rangle$, *i.e.* $S_0 = \emptyset$, $S_1 = S$ *and then* $l_1 = |B'|$, *which means that all the leading coefficients of polynomials in* B' *have been tested and are non-null on all of* $S = S_1$,
– *or* $\mathfrak{a}_0 \neq \langle 1\rangle$ *and then* $\mathfrak{a}_0 \supsetneq \mathfrak{a}$, $l_0 = l_1 - 1$.
(iii) P_1 *and* P_0 *are updated using the standard strategy and the Buchberger criterion of eliminating the pairs with disjoint set of variables of their* lpp.[3]

Proof. First of all we note that DiscussPolys will only be called with $l < |B|$. The list B' of polynomials is initially equal to B. The algorithm starts with testing if $B[l + 1]$ specializes to zero for every point of S. If this is the case we can simply delete $B[l + 1]$ from our list of polynomials and continue with considering the next polynomial $B[l + 2] = B'[l + 1]$ in the list. If we eventually find a polynomial which does not vanish identically on S, i.e. $f \neq 0$, then we use algorithm Split to test if there is an $a \in S$ with $\mathrm{lc}(f)(a) = 0$, i.e. $\mathfrak{a}_0 \neq \langle 1\rangle$. If this is the case we have found a proper splitting, and the two

[3] This is what is done in the present implementation, but this should be improved using better strategies as those developed in Gebauer and Möller (1988), Giovinni et al. (1991) and Faugère (2002).

Please cite this article in press as: Montes, A., Wibmer, M., Gröbner bases for polynomial systems with parameters. Journal of Symbolic Computation (2010), doi:10.1016/j.jsc.2010.06.017

Table 13
DiscussPolys algorithm.

```
(B', (𝔞_0, h_0), l_0, P_0, (𝔞_1, h_1), l_1, P_1) ← DiscussPolys((𝔞, h), B, l, P)
Input:
    ((𝔞, h), B, l, P): the current vertex data
Output:
    ((𝔞_0, h_0), B', l_0, P_0): a new vertex data making a new null assumption
    ((𝔞_1, h_1), B', l_1, P_1): a new vertex data making a new non-null assumption

begin
  B' := B
  split := false
  while split = false and l < |B'| do
    f := PNormalForm(B'[l + 1], (𝔞, h))
    if f = 0 then B' := B' with B'[l + 1] deleted
    else B' := B' with B'[l + 1] replaced by f
      ((𝔞_0, h_0), (𝔞_1, h_1)) := Split(lc(f), (𝔞, h))
      if 𝔞_0 ≠ ⟨1⟩ then split := true
      l_0 := l; l_1 := l + 1;
      P_1 := P ∪ {(j, l_1) : 1 ≤ j < l_1, (B[j], B[l_1]) ∈ Buchberger pair selection}
      P_0 := P
      else l := l + 1
      P := P ∪ {(j, l) : 1 ≤ j < l, (B[j], B[l]) ∈ Buchberger pair selection}
      end if
    end if
  end while
  if split = false then
    (𝔞_1, h_1) := (𝔞, h); (𝔞_0, h_0) := (⟨1⟩, h); l_0 := |B'|; l_1 := |B'|; P_0 := P; P_1 := P
  end if
end
```

appropriate new vertex data are returned. If $\mathrm{lc}(f)(a) \neq 0$ for all $a \in S$, i.e. $\mathfrak{a}_0 = \langle 1 \rangle$, then no splitting is necessary and we continue with the next polynomial in the list.

If it happens that we reach the end of the list then we must have *split* $=$ *false* and the last "if"-statement guarantees that we get back the correct result.

So (i) is a direct consequence of Proposition 19 and the remaining claims are immediate from the algorithm. □

The algorithm DiscussSPolys has some similarities with DiscussPolys (see Table 14). However DiscussPolys is always called with a vertex data $((\mathfrak{a}, h), B, l, P)$ satisfying $l < |B|$ whereas the vertex data for DiscussSPolys always satisfies $l = |B|$. In other words if DiscussSPolys is called with vertex data $((\mathfrak{a}, h), B, l, P)$ then $\mathrm{lc}(p)(a) \neq 0$ for all $p \in B$ and $a \in S = S((\mathfrak{a}, h))$. The task of DiscussSPolys is simply to carry on with the usual Buchberger algorithm until the next splitting is necessary, i.e until we encounter a leading coefficient which vanishes on some but not at all points of S.

The action of DiscussSPolys is summarized in Table 14.

Lemma 32 (*DiscussSPolys Algorithm*)**.** *Suppose that* DiscussSPolys *is called with the vertex data* $((\mathfrak{a}, h), B, l, P)$. *Then two new vertex data* $((\mathfrak{a}_0, h_0), B', l_0, P_0)$ *and* $((\mathfrak{a}_1, h_1), B', l_1, P_1)$ *with the following properties are obtained:*

(i) $S = S_0 \uplus S_1$ *where* $S = S((\mathfrak{a}, h))$, $S_0 = S((\mathfrak{a}_0, h_0))$ *and* $S_1 = S((\mathfrak{a}_1, h_1))$,
(ii) – *Either* $\mathfrak{a}_0 = \langle 1 \rangle$, *i.e.* $S_0 = \emptyset$, $S_1 = S$ *and then* $\mathrm{lc}(p)(a) \neq 0$ *for all* $a \in S = S_1$ *and* $p \in B'$. *Also* $P_1 = \emptyset$, *so that all the* S*-polynomials of pairs of elements of* B' *reduce to zero over* $S = S_1$. *In particular* B' *specializes to a Gröbner basis of* I_a *for every* $a \in S$.
– *Or* $\mathfrak{a}_0 \neq \langle 1 \rangle$ *and then* $\mathfrak{a}_0 \supsetneq \mathfrak{a}$, $l_0 = |B'| - 1$, $l_1 = |B'|$.
(iii) P_1 *and* P_0 *are updated using the current strategies.*

Proof. We recall that $\mathrm{lc}(p)(a) \neq 0$ for all $p \in B$ and $a \in S = S((\mathfrak{a}, h))$. The algorithm starts with picking a pair of polynomials of $B' = B$ specified in $P_1 = P$. This pair is removed from the list and we test if the reduction of the corresponding S-polynomial modulo B' vanishes identically on S, i.e. if

Table 14
DiscussSPolys algorithm.

```
(B', (a_0, h_0), l_0, P_0, (a_1, h_1), l_1, P_1) ← DiscussSPolys((a, h), B, l, P)
Input:
      (B, (a, h), l, P): the current vertex data
Output:
      ((a_0, h_0), B', l_0, P_0): a new vertex data making a new null assumption
      ((a_1, h_1), B', l_1, P_1): a new vertex data making a new non-null assumption

begin
   B' := B; P_1 := P
   split := false
   while split = false and P_1 ≠ ∅ do
      Pick (i, j) ∈ P_1  #{standard choice}
      P_1 := P_1 \ {(i, j)}
      f := lc(B[j])B[i] − lc(B[i])B[j]
      f := PNormalForm(Pdiv(f, B'), (a, h))
      if f ≠ 0 then
         B' := B' ∪ {f}
         ((a_0, h_0), (a_1, h_1)) := Split(lc(f), (a, h))
         if a_0 ≠ ⟨1⟩ then split := true
            l_0 := |B'| − 1; P_0 := P_1
            l_1 := |B'|; P_1 := P_1 ∪ {(j, l_1) : 1 ≤ j < l_1, (B[j], f) ∈ BPS}
         else l := l + 1; P_1 := P_1 ∪ {(j, l) : 1 ≤ j < l, (B[j], f) ∈ BPS}
         end if
      end if
   end while
   if split = false then
      (a_1, h_1) := (a, h); (a_0, h_0) := (⟨1⟩, h); l_0 := |B'|; l_1 := |B'|; P_0 := ∅; P_1 := ∅
   end if
end
```

$f = 0$. If this is the case the algorithm continues by picking the next pair from P_1. Otherwise, i.e. if $f \neq 0$ we add f to the basis and use algorithm Split to test if there is an $a \in S$ with $\mathrm{lc}(f)(a) = 0$, i.e. $\mathfrak{a}_0 \neq \langle 1 \rangle$. If this is the case we have found a proper splitting, and the two appropriate new vertex data are returned. If $\mathrm{lc}(f)(a) \neq 0$ for all $a \in S$, i.e. $\mathfrak{a}_0 = \langle 1 \rangle$, then no splitting is necessary and we continue by picking the next pair in P_1.

If it happens that we remove the last element from P_1 then we must have *split* $=$ *false* and the last "if"-statement guarantees that we return the correct result. We note that only in this case we will have $\mathfrak{a}_0 = \langle 1 \rangle$. That the list P_1 is empty means that for each pair from the current basis $B' = \{p_1, \ldots, p_r\}$ the corresponding S-polynomial reduces to zero modulo B' over S. In other words for every $a \in S$ the polynomials $\{p_1(a, \overline{x}), \ldots, p_r(a, \overline{x})\}$ satisfy Buchberger's criterion and thus are a Gröbner basis of I_a. $\square$

Finally, we give the details for algorithm ReduceGB. It is the obvious generalization of the final steps in the usual Buchberger algorithm. It is described in Table 15. First it minimizes the Gröbner basis and then fully reduces the minimized Gröbner basis.

3.3. *Computing the bases*

The last main step in algorithm GCover is Basis. The algorithm Basis determines generic representations of the monic I-regular functions in the bases of the canonical Gröbner cover. It is called by GCover for each lpp-segment.

When BuildTree has finished GCover has already obtained a finite partition of $\overline{K}^m$ into parametric subsets $S_1, \ldots, S_s$ and bases $B_1, \ldots, B_s \subset K[\overline{a}, \overline{x}]$ such that $\mathrm{lpp}(B_i)$ is the minimal generating set of $\mathrm{lpp}(S_i)$ and evaluating B_i at $a \in S_i$ yields the reduced Gröbner basis of I_a (up to normalization) for $i = 1, \ldots, s$.

ARTICLE IN PRESS

Table 15
ReduceGB algorithm.

```
B' ← ReduceGB(B)
Input: B: a finite subset of K[ā][x̄] such that for every a in a certain locally closed subset S of K̄^m
       we have lc(p)(a) ≠ 0 for all p ∈ B and B(a) ⊂ K̄[x̄] is a Gröbner basis.
Output: B': a finite subset of K[ā][x̄] such that B'(a) is (up to normalization)
       the reduced Gröbner basis of ⟨B(a)⟩ ⊂ K̄[x̄] for every a ∈ S.

begin
  Let B' ⊂ B be the set of all polynomials in B with minimal lpp.
  for p ∈ B' do
    B' := B' ∖ {p}
    p := PDiv(p, B')
    B' := B' ∪ {p}
  end do
end
```

The next step is to compute the lpp-segments (see Theorem 8). For a fixed occurring set T of leading power products the corresponding lpp-segment

$$S = \bigcup_{\mathrm{lpp}(S_i)=T} S_i$$

is computed with algorithm LCUnion which was already explained in Section 2.1.1. If $\mathfrak{p}_1, \ldots, \mathfrak{p}_r$ are the components of S (see Definition 14) then for each $i \in \{1, \ldots, r\}$ there exists a unique $j = j(i)$ such that $\mathrm{lpp}(S_j) = T$ and S_j has $\mathfrak{p}_i$ as component (cf. the beginning of the proof of Proposition 20). This S_j is already determined by LCUnion. The input for algorithm Basis then is

$$((\mathfrak{p}_1, B_{j(1)}), \ldots, (\mathfrak{p}_r, B_{j(r)})).$$

Proposition 33 (*Basis Algorithm*)**.** *Let $I \subset K[\overline{a}][\overline{x}]$ be a homogeneous ideal and S an* lpp-segment *with respect to I. Then algorithm* Basis *(see Table 16) computes generic representations of the elements in the reduced Gröbner basis of I over S.*

Proof. From the theoretical point of view the while loop in algorithm Basis is not necessary. Algorithm Combine (see Table 8) would give the desired result in any case. So we only need to explain the while loop.

As in the algorithm fix $t \in T$ and for $i = 1, \ldots, r$ let $p_i \in B_i$ denote the unique element of B_i with $\mathrm{lpp}(p_i) = t$. Let f denote the monic I-regular function in the reduced Gröbner basis of I over S with $\mathrm{lpp}(f) = t$. The purpose of the while loop is simply to test if already one of the p_i's is a generic representation of f. Fix $i \in \{1, \ldots, r\}$.

We claim that p_i is a generic representation of f if and only if for each $j \in \{1, \ldots, r\}$ we have $\mathrm{lc}(p_i) \notin \mathfrak{p}_j$ and the coefficients of $\mathrm{lc}(p_i)p_j - \mathrm{lc}(p_j)p_i$ lie in $\mathfrak{p}_j$.

But $\mathrm{lc}(p_i) \notin \mathfrak{p}_j$ for $j = 1, \ldots, r$ is equivalent to saying that $S \setminus \mathbb{V}(\mathrm{lc}(p_i))$ is dense in S and that the coefficients of $\mathrm{lc}(p_i)p_j - \mathrm{lc}(p_j)p_i$ lie in $\mathfrak{p}_j$ means that

$$\frac{p_i(a, \overline{x})}{\mathrm{lc}(p_i(a))} = \frac{p_j(a, \overline{x})}{\mathrm{lc}(p_j(a))} = f(a)$$

for every $a \in S \cap \mathbb{V}(\mathfrak{p}_j) \setminus \mathbb{V}(\mathrm{lc}(p_i)\mathrm{lc}(p_j))$ and $j = 1, \ldots, r$. Thus the claim is immediate from Definition 24. □

4. The GröbnerCover algorithm

In this section we present our main algorithm GröbnerCover (see Table 17). It takes as input a finite generating set of the ideal $I \subset K[\overline{a}, \overline{x}]$ (and of course the term-order $\succ_{\overline{x}}$ on the variables) and computes the canonical Gröbner cover of $\overline{K}^m$ with respect to I and $\succ_{\overline{x}}$ (Definition 11). The monic I-regular functions in the bases are given in full representation. The ideal I need not be homogeneous

Please cite this article in press as: Montes, A., Wibmer, M., Gröbner bases for polynomial systems with parameters. Journal of Symbolic Computation (2010), doi:10.1016/j.jsc.2010.06.017

Table 16
Basis algorithm.

$B \leftarrow$ **Basis**(H)
Input:
$H = ((\mathfrak{p}_1, B_1), \ldots, (\mathfrak{p}_r, B_r))$: The $\mathfrak{p}_i$'s are pairwise distinct prime ideals of $K[\overline{a}]$ and they are the components of an lpp-segment S. The B_i's are subsets of $K[\overline{a}][\overline{x}]$ all having the same lpp T.
Output:
B: a finite subset of $K[\overline{a}][\overline{x}]$ with $\mathrm{lpp}(B) = T$ and such that each element of B is a generic representation of the corresponding element in the Gröbner Basis of I over S

begin
  $B := \emptyset$
  for each $t \in T$ **do**
    For $i = 1, \ldots, r$ let p_i denote the polynomial of B_i with $\mathrm{lpp}(p_i) = t$.
    $i := 1$; *generic* := *false*
    while *generic* = *false* and $i \leq r$ **do**
      if $\mathrm{lc}(p_i) \notin \mathfrak{p}_j$ and the coefficients of $\mathrm{lc}(p_i)p_j - \mathrm{lc}(p_j)p_i$ lie in $\mathfrak{p}_j$
        for $j = 1, \ldots, r$
      then *generic* := *true*; $p := p_i$
      end if
      $i := i + 1$
    end while
    if *generic* = *false* **then**
      $p :=$ Combine$(((\mathfrak{p}_1, p_1), \ldots (\mathfrak{p}_r, p_r)))$
    end if
    $B := B \cup \{p\}$
  end do
end

Table 17
Algorithm GröbnerCover.

$G \leftarrow$ **GröbnerCover**$(F, \succ_{\overline{x}})$
Input:
F: a finite generating set of the ideal $I \subset K[\overline{a}][\overline{x}]$
Output:
G: the canonical Gröbner cover of $\overline{K}^m$ with respect to I

begin
  if all the elements in F are homogeneous **then**
    $((S_1, B_1), \ldots, (S_r, B_r)) :=$ GCover $(F, \succ_{\overline{x}})$
  else
    let $f_1, \ldots, f_s \in K[\overline{a}][\overline{x}, x_0]$ be a generating set of the homogenization of I
    $((S_1, B_1), \ldots, (S_r, B_r)) :=$ GCover $\left(\{f_1, \ldots, f_s\}, \succ_{\overline{x},x_0}\right)$
    for $i = 1, \ldots, r$ **do**
      $B_i :=$ ReduceGB$(B_i(\overline{x}, 1))$
    end do
  end if
  $G := \emptyset$
  for $i = 1, \ldots, r$ **do** $B := \emptyset$
    for $p \in B_i$ **do**
      $B := B \cup \{$ExtendPoly$(S_i, p)\}$
    end do
    $G := G \cup \{(S_i, B)\}$
  end do
end

but nevertheless GröbnerCover will distinguish the two cases whether or not the generators are homogeneous.

If the generators are homogeneous then GröbnerCover calls algorithm Gcover (see Table 10) to obtain the canonical Gröbner cover. In this case it only remains to convert the generic representations

ARTICLE IN PRESS

Table 18
Algorithm ExtendPoly.

$q \leftarrow$ **ExtendPoly**(S, p)
```
Input:
```
S: a locally closed subset of $\overline{K}^m$
$p = \sum_\alpha p_\alpha \overline{x}^\alpha \in K[\overline{a}][\overline{x}]$: a generic representation of a monic I-regular function f on S
```
Output:
```
q: a full representation of f on S

begin
Let $(\mathfrak{a}, \mathfrak{b})$ be the C-representation of S.
if $\mathfrak{b} \subseteq \sqrt{\mathfrak{a} + \langle \mathrm{lc}(p) \rangle}$ **then** $q := \sum_\alpha (p_\alpha; \mathrm{lc}(p))\overline{x}^\alpha$
else
$q := \sum_\alpha$ Extend$(S, p_\alpha, \mathrm{lc}(p))\overline{x}^\alpha$
end if
end

of the basis elements given by Gcover into full representations. This is done by algorithm ExtendPoly (see Table 18).

If not all the generators are homogeneous we first need to compute the homogenization J of I. Then we apply Gcover to a finite generating set of J and obtain the canonical Gröbner cover of $\overline{K}^m$ with respect to J. By definition the segments of the canonical Gröbner cover with respect to I are the segments of the canonical Gröbner cover with respect to J. And the bases in the canonical Gröbner cover with respect to I are obtained from the bases in the canonical Gröbner cover with respect to J by dehomogenizing, minimizing and reducing (as demonstrated in the proof of Proposition 10). Thus we only have to apply algorithm ReduceGB (see Table 15) to obtain the generic representations of the basis elements in the canonical Gröbner cover with respect to I. As in the homogeneous case we apply ExtendPoly in the end to obtain full representations (Table 16).

The GröbnerCover algorithm is given in Table 17.

4.1. The case of arbitrary ideals

As explained above, if the ideal I is not homogeneous then algorithm GröbnerCover will need to compute its homogenization. The purpose of this short subsection is to show how this can be done. Throughout this subsection we suppose that $I \subset K[\overline{a}][\overline{x}]$ is an arbitrary ideal and as always we also have a fixed monomial order $\succ_{\overline{x}}$ on the variables. As in Section 1 we consider the ring $K[\overline{a}][\overline{x}, x_0]$ with the extended monomial order $\succ_{\overline{x},x_0}$ defined by

$$\overline{x}^\alpha x_0^d \succ_{\overline{x},x_0} \overline{x}^\beta x_0^e$$

if $\overline{x}^\alpha \succ_{\overline{x}} \overline{x}^\beta$ or $\overline{x}^\alpha = \overline{x}^\beta$ and $d > e$. For a polynomial $P \in K[\overline{a}][\overline{x}]$ we denote with $\deg(P)$ its total degree with respect to $\overline{x}$ and with $\eta(P) \in K[\overline{a}][\overline{x}, x_0]$ its homogenization, i.e. $\eta(P) = x_0^{\deg(P)} P\left(\frac{x_1}{x_0}, \ldots, \frac{x_n}{x_0}\right)$. With J we denote the homogenization of I, i.e.

$$J = \langle \eta(P) : P \in I \rangle \subset K[\overline{a}][\overline{x}, x_0].$$

Proposition 34 (*Basis of Homogenization*)**.** *Let $I \subset K[\overline{a}][\overline{x}]$ be an arbitrary ideal, $>_{\overline{x}}$ a graded term-order on $\overline{x}$ and $>_{\overline{x},\overline{a}}$ a product order considering also the parameters $\overline{a}$ as variables. If $g_1, \ldots, g_m$ is a Gröbner basis of I with respect to $>_{\overline{x},\overline{a}}$. Then $\eta(g_1), \ldots, \eta(g_m)$ is a generating set of the homogenization $J \subset K[\overline{a}][\overline{x}, x_0]$ of I.*

Proof. Let $g \in I \subset K[\overline{a}, \overline{x}]$. Since $g_1, \ldots, g_m$ is a Gröbner basis there exist polynomials $f_1, \ldots, f_m \in K[\overline{a}, \overline{x}]$ such that $g = f_1 g_1 + \cdots + f_m g_m$ with $\mathrm{lpp}_{\overline{x},\overline{a}}(g) \leq_{\overline{x},\overline{a}} \mathrm{lpp}_{\overline{x},\overline{a}}(f_i g_i)$ for every i. Since $>_{\overline{x},\overline{a}}$ is a product order this implies $\mathrm{lpp}_{\overline{x}}(g) \leq_{\overline{x}} \mathrm{lpp}_{\overline{x}}(f_i g_i)$ for every i, and thus, $>_{\overline{x}}$ being a graded order also

Please cite this article in press as: Montes, A., Wibmer, M., Gröbner bases for polynomial systems with parameters. Journal of Symbolic Computation (2010), doi:10.1016/j.jsc.2010.06.017

$\deg(f_i g_i) \le d = \deg(g)$. Therefore

$$\begin{aligned}\eta(g) &= x_0^{d-\deg(f_1g_1)}\eta(f_1g_1) + \cdots + x_0^{d-\deg(f_mg_m)}\eta(f_mg_m)\\ &= x_0^{d-\deg(f_1g_1)}\eta(f_1)\eta(g_1) + \cdots + x_0^{d-\deg(f_mg_m)}\eta(f_m)\eta(g_m)\\ &\in \langle \eta(g_1), \ldots, \eta(g_m)\rangle.\end{aligned}$$

Consequently $J = \langle \eta(g_1), \ldots, \eta(g_m)\rangle$. □

4.2. *The* ExtendPoly *algorithm*

The task of the ExtendPoly algorithm is to convert a generic representation of a monic I-regular function into a full representation.

So let $S \subset \overline{K}^m$ be a locally closed subset, $f : S \to \overline{K}[\overline{x}]$ a monic I-regular function and $p = \sum_\alpha p_\alpha \overline{x}^\alpha \in K[\overline{a}][\overline{x}]$ a generic representation of f (see Definition 24). Generic representations are very practical to handle on the computer and allow us to manipulate with monic I-regular function easily, however they have the drawback that the value $f(a)$ of f at a point of $a \in S$ cannot immediately be determined if $\mathrm{lc}(p)(a) = 0$. This is why ExtendPoly is applied at the very end in GröbnerCover algorithm.

If $\mathrm{lc}(p)(a) \neq 0$ for all $a \in S$ there is no need to take action, and formally the polynomial $\sum_\alpha (p_\alpha; \mathrm{lc}(p))\overline{x}^\alpha$ is a full representation of f. Otherwise we simply apply Extend algorithm to the coefficients: We know that $(p_\alpha; \mathrm{lc}(p))$ is a generic representation of $\mathrm{coef}(f, \alpha) \in \mathcal{O}(S)$ and so Extend$(S, p_\alpha, \mathrm{lc}(p))$ provides a full representation of $\mathrm{coef}(f, \alpha)$ and

$$\sum_\alpha \textsc{Extend}(S, p_\alpha, \mathrm{lc}(p))\overline{x}^\alpha$$

is a full representation of f.

To test if $\mathrm{lc}(p)(a) \neq 0$ for all $a \in S$ we can use the following simple lemma.

Lemma 35. *Let $q \in K[\overline{a}]$, $\mathfrak{a}$, $\mathfrak{b}$ ideals of $K[\overline{a}]$ and $S = \mathbb{V}(\mathfrak{a}) \setminus \mathbb{V}(\mathfrak{b})$. Then $q(a) \neq 0$ for all $a \in S$ if and only if*

$$\mathfrak{b} \subseteq \sqrt{\mathfrak{a} + \langle q\rangle}.$$

Proof. $q(a) \neq 0$ for all $a \in S$ if and only if $\mathbb{V}(q) \cap S = \emptyset$. We have:

$$\begin{aligned}\mathbb{V}(q) \cap S &= \emptyset \Leftrightarrow \mathbb{V}(q) \cap (\mathbb{V}(\mathfrak{a}) \setminus \mathbb{V}(\mathfrak{b})) = \emptyset \Leftrightarrow \mathbb{V}(\mathfrak{a}) \cap \mathbb{V}(q) \cap (\mathbb{V}(\mathfrak{a}) \setminus \mathbb{V}(\mathfrak{b})) = \emptyset\\ &\Leftrightarrow \mathbb{V}(\mathfrak{a} + \langle q\rangle) \cap (\mathbb{V}(\mathfrak{a}) \setminus \mathbb{V}(\mathfrak{b})) = \emptyset \Leftrightarrow \mathbb{V}(\mathfrak{a} + \langle q\rangle) \subseteq \mathbb{V}(\mathfrak{b}) \Leftrightarrow \mathfrak{b} \subseteq \sqrt{\mathfrak{a} + \langle q\rangle}.\end{aligned}$$ □

The correctness of ExtendPoly algorithm given in Table 18 is immediate from the above explanations.

4.3. *Some remarks on implementation issues*

When presenting our algorithms in this article we have tried to keep things as simple as possible. Our goal was to clearly state what the algorithm does without giving too much technical details. For the sake of a clear exposition and to keep this paper at a reasonable length we have sometimes left out improvements that are present in the actual implementation. The purpose of this subsection is to give some hints on this improvements and to give some insights into the practical performance of the GröbnerCover algorithm.

A critical aspect for the efficiency of the whole GröbnerCover algorithm is the use of primary decomposition, that is essential in every algorithm that tries to obtain a canonical discussion of parametric polynomial systems. At this effect, it should be noted, that in the first BuildTree part of the algorithm where most of the computation is done, the incremental algorithms RrepNN and

RrepN avoid the complete use of primary decomposition, and only simple incremental radicals are used (in RrepN). Only after BuildTree is finished, the R-representations must be transformed into P-representations, and then the routine RtoPrep involves primary decomposition. An appropriate design of the special primary decompositions involved there is mandatory for effectiveness.

There is another critical problem inside Buildtree, namely the computation of the "generic" case, i.e. when the algorithm follows the path to the left most terminal vertex making only new non-null assumptions. There is some work in progress to speed up the computation in the generic case.

For example, when the generic basis is $\{1\}$, and this is usual in automatic theorem discovering, we can use an alternative strategy. Computing the Gröbner basis with respect to the product of a graded order in $\overline{x}$ and an order in the parameters (what is needed to compute the homogenized ideal) we obtain also the elimination ideal in the parameters $I_0 = I \cap K[\overline{a}]$. If I_0 is non-null, then the generic basis is $\{1\}$ and the generic segment can be obtained, in P-representation by simply compute the prime decomposition of I_0, and taking the whole parameter space minus $V(I_0)$. Let $\{\mathfrak{p}_1, \ldots, \mathfrak{p}_r\}$ be the minimal primes of I_0. Then, we can compute separately the particular trees for each of the components with the restriction of $\mathbb{V}(\mathfrak{p}_i)$, which will be much simpler to do, and then summarize the result.

We note that in the implementation one can optionally specify a certain locally closed subset S of $\overline{K}^m$ and then GröbnerCover will only compute the canonical Gröbner cover of S.

Practical experiments show that if the generators $p_1, \ldots, p_r$ of our ideal I under consideration are not homogeneous then Buildtree applied to generators of the homogenization of I usually has a much longer running time then Buildtree applied to $p_1, \ldots, p_r$. This seems to be due mainly to the fact that in general one has many generators of the homogenization of I.

Thus in computationally hard problems it is recommended to avoid the computation of the homogenization but to simply apply our algorithms to the homogenizations $\eta(p_1), \ldots, \eta(p_r)$ of $p_1, \ldots, p_r$. We note that Buildtree$(p_1, \ldots, p_r)$ and Buildtree$(\eta(p_1), \ldots, \eta(p_r))$ essentially perform the same computations. This way one is not guaranteed to obtain the canonical Gröbner cover with respect to I but the result will be reasonably simple.

Concerning memory consumption we remark that it is not necessary that algorithm Buildtree stores the vertex data of intermediate (i.e. non-terminal) vertices. This has been done historically for didactic purposes, but it is unnecessary.

The algorithm Combine tends to produce rather complicated polynomials but one can always reduce them modulo $\mathfrak{a}$ where $\mathfrak{a} \subset K[\overline{a}]$ is the radical ideal with $S = \mathbb{V}(\mathfrak{a})$ and S is the locally closed set over which we are working. In algorithm Combine one can collect together all the components of the lpp-segment which are coming from the same Buildtree segment to simplify and speed up the computation.

On the contrary Extend often produces quite simple polynomials which sometimes are even simpler and more "generic" then those originally found by Buildtree. For example it might happen that on a certain lpp-segment S none of the polynomials found by Buildtree gives the correct value on all points of S but with Extend respectively ExtendPoly we are able to obtain a polynomial with this property (cf. Examples 27, 29 and example in Section 5).

One could also consider the possibility of replacing Buildtree with an alternative algorithm such as Suzuki–Sato Algorithm (Suzuki and Sato, 2006) in case Buildtree is not able to finish within reasonable time. One would only need to transform the output of Suzuki–Sato algorithm into a disjoint reduced comprehensive Gröbner system to be able to apply our algorithms.

The full representation of an I-regular function as given in Definition 22 is a bit awkward to handle in a computer algebra system. One can use instead the representation given in the following definition.

Definition 36 (*Complete Representation*). Let $S \subset \overline{K}^m$ be locally closed and $f : S \to \overline{K}[\overline{x}]$ a monic I-regular function. Let $p_1, \ldots, p_r \in K[\overline{a}][\overline{x}]$. We say that $(p_1, \ldots, p_r)$ is a *complete representation of f* if

(i) $f(a) = \frac{p_i(a,\overline{x})}{\mathrm{lc}(p_i)(a)}$ for every $a \in S$ with $\mathrm{lc}(p_i)(a) \neq 0$,
(ii) for every $a \in S$ there exists $i \in \{1, \ldots, r\}$ such that $\mathrm{lc}(p_i)(a) \neq 0$ and
(iii) $\mathrm{lc}(p_i)(a)p_j(a, \overline{x}) = \mathrm{lc}(p_j)(a)p_i(a, \overline{x})$ for all $a \in S$ and $1 \leq i, j \leq r$.

We note that (ii) and (iii) imply that $p_i(a, \overline{x}) = 0$ for $a \in S$ with $\mathrm{lc}(p_i)(a) = 0$.

Please cite this article in press as: Montes, A., Wibmer, M., Gröbner bases for polynomial systems with parameters. Journal of Symbolic Computation (2010), doi:10.1016/j.jsc.2010.06.017

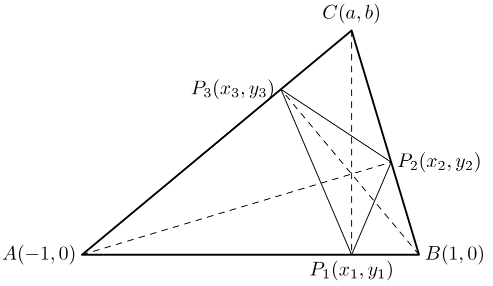

Fig. 1. Orthic triangle.

From the theoretical point of view the usage of I-regular functions instead of just polynomials in $K[\overline{a}][\overline{x}]$ is very important. The results about I-regular functions in the first section are needed to establish the main algorithms in the sections three and four. However in the practical examples it appears that most of the time the monic I-regular functions in the bases of the canonical Gröbner cover can be completely represented by a single polynomial, although it is not difficult to construct examples where several polynomials will be needed.

There is an obvious way of converting a full representation into a complete representation by clearing denominators. This seems to create a large number of polynomials in the complete representation, since one has to consider all possible combinations. However we can drastically reduce this number. It suffices to take a subset $\{p_1, \dots, p_r\}$ with the property that for every $a \in S$ there exists $i \in \{1, \dots, r\}$ with $\mathrm{lc}(p_i)(a) \neq 0$, i.e. $\mathbb{V}(\langle \mathrm{lc}(p_1), \dots, \mathrm{lc}(p_r)\rangle) \cap S = \emptyset$, or equivalently $\mathfrak{b} \subset \sqrt{\mathfrak{a} + \langle \mathrm{lc}(p_1), \dots, \mathrm{lc}(p_r)\rangle}$. One can also attempt to find such subsets by using the segments obtained by BUILDTREE.

In this way one usually never finds more then two or three polynomials in a complete representation in the final output of GRÖBNERCOVER, (except in examples which have been cooked up for this purpose).

5. Example

To fix ideas, let us give an application using the GRÖBNERCOVER algorithm. We present the problem, the concise answer obtained by the algorithm and its geometrical interpretation. We also comment on the complexity of the computations during the algorithm.

We consider the following problem: Find the points $C = (a, b)$ on the plane for which the triangle ABC of Fig. 1 has an orthic triangle (the triangle $P_1P_2P_3$ through the foots of the heights) that is isosceles (with sides $\overline{P_1P_2} = \overline{P_1P_3}$).

We have $P_1 = (a, 0)$. Joining the equations defining the points P_2 and P_3 and the condition for the orthic triangle to be isosceles, we have the following ideal representing the system of equations:

$$\begin{aligned} I \;=\; &\langle (a-1)y_2 - b(x_2-1),\ (a-1)(x_2+1) + by_2,\\ &(a+1)y_3 - b(x_3+1),\ (a+1)(x_3-1) + by_3,\\ &(x_3-a)^2 + y_3^2 - (x_2-a)^2 - y_2^2\rangle. \end{aligned}$$

Applying the full GRÖBNERCOVER algorithm, using $\succ_{\overline{x}} = \mathrm{grevlex}(x_2, x_3, y_2, y_3)$, we obtain the following very concise result:

1. Segment with lpp $= \{1\}$	Generic segment
Basis: $\{1\}$.	
P-representation of the segment: $\big(\langle 0\rangle, (\langle a^2 - b^2 - 1\rangle, \langle a^2 + b^2 - 1\rangle, \langle a\rangle)\big)$.	

2. Segment with lpp $= \{y_3, y_2, x_3, x_2\}$

Basis:

$$\{(a^2+b^2+2a+1)y_3+(-2ab-2b),\\ (a^2+b^2-2a+1)y_2+(2ab-2b),\\ (a^2+b^2+2a+1)x_3+(-a^2+b^2-2a-1),\\ (a^2+b^2-2a+1)x_2+(a^2-b^2-2a+1)\}.$$

P-representation of the segment:

$$\big(\langle a^2+b^2-1\rangle, (\langle b, a-1\rangle, \langle b, a+1\rangle)\big);\\ \big(\langle a^2-b^2-1\rangle, (\langle b, a-1\rangle, \langle b, a+1\rangle, \langle b^2+1, a\rangle)\big);\\ \big(\langle a\rangle, (\langle b^2+1, a\rangle)\big)$$

3. Segment with lpp $= \{y_3, x_3, x_2^2\}$

Basis: $\{y_3,\ x_3-1,\ x_2^2+y_2^2-2x_2+1\}$.

P-representation of the segment: $(\langle b, a-1\rangle, (\langle 1\rangle))$

4. Segment with lpp $= \{1\}$

Basis: $\{1\}$.

P-representation of the segment: $\big(\langle b^2+1, a\rangle, (\langle 1\rangle)\big)$

5. Segment with lpp $= \{y_2, x_2, x_3^2\}$

Basis: $\{y_2, x_2+1, x_3^2+y_3^2+2x_3+1\}$.

P-representation of the segment: $(\langle b, a+1\rangle, (\langle 1\rangle))$.

We observe that there are only 5 segments in the canonical Gröbner cover and the single repeated lpp corresponds to segments 1 and 4.

The bases of the segments 1 and 4 are $\{1\}$, showing that there does not exists any solution in those segments. The important segment for our problem is segment 2 with lpp $= \{y_3, y_2, x_3, x_2\}$ (i.e. the set of variables), as it shows that in this segment it exists a unique solution for the points P_2 and P_3 (that are determined by the basis). We obtain three branches of the solution, namely

(1) $a = 0$
(2) $a^2 + b^2 - 1 = 0$
(3) $a^2 - b^2 - 1 = 0$

except the points $A = (-1, 0)$ and $B = (1, 0)$ corresponding to degenerate triangles, and two complex points $M = (i, 0)$, $N = (-i, 0)$. Branch (1) represents isosceles triangles and is an obvious solution. Branch (2) (circle) represents rectangular triangles for which the orthic triangle is isosceles with basis of length 0 and is also obvious. But branch (3) gives points on a hyperbola for which the given triangle ABC is neither isosceles nor rectangle but has an orthic triangle that is isosceles and is not an obvious solution.

Segments 3 and 5 correspond respectively to the degenerate triangles with $C = A = (1, 0)$ and $C = B = (-1, 0)$. Finally segment 4 represents the two imaginary points $C = M(0, i)$ and $C = N(0, -i)$ for which no solution exists as for the points in segment (1), but these points are not summarized into a single segment by the canonical Gröbner cover. The fundamental reason for this is that they come from two segments of the homogenized ideal with different lpp. We also remark that the union of segment 1 and segment 4 is not locally closed. This is another good reason why the canonical Gröbner cover does not summarize them into a single segment.

Let us now give some clarifying details about the development of the algorithm and its complexity. Even if the final output of the discussion with GröbnerCover is very simple and concise, the computations to obtain it are not so simple. In fact, we choose this example because all the resources of the powerful algorithm are used.

First of all, the given ideal I is non-homogeneous. So to compute the canonical Gröbner cover we first need to homogenize it. To compute the homogenization J of I we need a graded order in the variables. We use the graded order $>_{\overline{x}} = \text{grevlex}(x_2, x_3, y_2, y_3)$ and grevlex(a, b) for the parameters. We must first compute a Gröbner basis of I with respect to the product order $(>_{\overline{x}} \cdot \text{grevlex}(a, b))$ and then homogenize it using the new variable x_0. The result is a basis with 22 homogeneous polynomials.

Now begins the algorithm GCover for homogeneous ideals. We must now use the product order of $\succ_{\overline{x}}$ (that we take also to be $\succ_{\overline{x}} = \text{grevlex}(x_2, x_3, y_2, y_3)$) and grevlex$(x_0)$, resulting in $\succ_{\overline{x},x_0} = (\succ_{\overline{x}} \cdot \text{grevlex}(x_0))$. We could also use another discussion order $\succ_{\overline{x}}$, for example lex(x_2, x_3, y_2, y_3), but we expect that the discussion will be simpler with this choice. We apply BuildTree, then select the terminal vertices, group them by lpp and transform the reduced representations of the segments into P-representations.

BuildTree obtains 16 little segments for the first lpp-segment of the canonical Gröbner cover with lpp $= \{1\}$, 7 little segments for the second lpp-segment with lpp $= \{y_3, y_2, x_3, x_2\}$ and a single segment for each of the three remaining lpp-segments. The fourth lpp-segment having lpp $= \{t, y_2^2, x_3, x_2\}$ reduces to basis $\{1\}$ after dehomogenization producing two final segments with lpp $= \{1\}$. BuildTree also obtains full representations of the bases for segments 1, 3, 4, 5, and the algorithm does not need to use neither Combine nor Extend algorithm for these.

LCUnion must be used to compute the P-representation of the union of the 16 respectively 7 little segments obtained by BuildTree. The result is the simple description of the final output given above.

Now let us detail what happens with the bases in the 7 little segments forming segment 2 of the canonical Gröbner cover with lpp $= \{y_3, y_2, x_3, x_2\}$. The segment has three components, corresponding to $\mathfrak{p}_1 = \langle a^2 + b^2 - 1\rangle$, $\mathfrak{p}_2 = \langle a^2 - b^2 - 1\rangle$ and $\mathfrak{p}_3 = \langle a\rangle$, with bases obtained by BuildTree as follows: Basis $B_1 = \{p_1, p_2, p_3, p_4\}$ for $\mathfrak{p}_1$ and $\mathfrak{p}_2$ where

$$\begin{aligned}
p_1 &= 2b(2a + b^2 + 1)y_3 + (a^3 + a^2b^2 - a^2 - 3ab^2 - a - b^4 - 4b^2 + 1)x_0,\\
p_2 &= 2b(2a + b^2 + 1)y_2 + (3a^3 + a^2b^2 + 3a^2 - ab^2 - 3a - b^4 - 3)x_0,\\
p_3 &= 2(2a + b^2 + 1)x_3 + (a^3 - 2a^2 - ab^2 - 3a + 2b^2 - 2)x_0,\\
p_4 &= 2(2a + b^2 + 1)x_2 + (a^3 - 2a^2 - ab^2 - 3a + 2b^2 - 2)x_0,
\end{aligned}$$

and $B_2 = \{q_1, q_2, q_3, q_4\}$ for $\mathfrak{p}_3$, where

$$\begin{aligned}
q_1 &= (b^2 + 1)y_3 + (-2b)x_0,\\
q_2 &= (b^2 + 1)y_2 + (-2b)x_0,\\
q_3 &= (b^2 + 1)x_3 + (b^2 - 1)x_0,\\
q_4 &= (b^2 + 1)x_2 + (-b^2 + 1)x_0.
\end{aligned}$$

We shall only discuss what happens with the first polynomial of the bases, the other three having the same comportment. First the algorithm verifies that neither p_1 specializes to q_1 on an open set of $\mathbb{V}(\mathfrak{p}_3)$ nor q_1 specializes to p_1 on an open set of $\mathbb{V}(\mathfrak{p}_1 \cap \mathfrak{p}_2)$. So the algorithm continues applying:

$$\text{Combine}((\mathfrak{p}_1, p_1), (\mathfrak{p}_2, p_1), (\mathfrak{p}_3, q_1)) = h$$

where

$$\begin{aligned}
h = {} & (2a^5b^3 + 2a^5b + a^4b^5 + 6a^4b^3 + 5a^4b + 2a^3b^5 - 2a^3b - 2a^2b^5\\
& - 8a^2b^3 - 6a^2b - 2ab^7 - 4ab^5 - 2ab^3 - b^9 - 2b^7 + 2b^3 + b)y_3\\
& + (a^6b^2 + a^6 + a^5b^4 - 4a^5b^2 - a^5 - 5a^4b^4 - 7a^4b^2 - 2a^4 - a^3b^6\\
& - 6a^3b^4 + 5a^3b^2 + 2a^3 + 7a^2b^4 + 8a^2b^2 + a^2 + 5ab^6 + 5ab^4 - ab^2\\
& - a + 2b^8 + 2b^6 - 2b^4 - 2b^2)x_0,
\end{aligned}$$

is know to specialize well in an open and dense subset of $\mathbb{V}(\mathfrak{p}_1) \cup \mathbb{V}(\mathfrak{p}_2) \cup \mathbb{V}(\mathfrak{p}_3)$. Nevertheless one can verify that h reduces to zero on some points of the segment, so we will need to use Extend algorithm. But before this, we dehomogenize, minimize and reduce the bases.

Please cite this article in press as: Montes, A., Wibmer, M., Gröbner bases for polynomial systems with parameters. Journal of Symbolic Computation (2010), doi:10.1016/j.jsc.2010.06.017

Then we apply Extend on the corresponding segment. The result are 3 polynomials h_1, h_2, h_3, where

$$\begin{aligned} h_1 &= (a^2 + b^2 + 2a + 1)y_3 + (-2ab - 2b), \\ h_2 &= (2ab^2 - 2b^2 - 2a - 2)y_3 + (a^3b - ab^3 - 2a^2b + ab + 4b), \\ h_3 &= (-2b^3 - 4ab - 2b)y_3 + (a^4 - a^2b^2 - a^3 + 3ab^2 - a^2 + 4b^2 + a), \end{aligned}$$

that are known to form a full representation of the I-regular function on the whole segment. The algorithm continues analyzing for all the 6 little segments if the polynomials h_1, h_2, h_3 remain non-null on them. It realizes that h_1 alone is non-null on all the 6 little segments, so that h_2 and h_3 are unnecessary. Finally it outputs the full representation of the I-regular function f_1 consisting of the single polynomial h_1, even if Extend has been used.

Acknowledgements

The first author was partially supported by the Spanish Ministerio de Ciencia y Tecnología under project MTM2009-07242, and by the Generalitat de Catalunya under project 2009SGR1040. The second author was supported by GTEM, MRTN-CT-2006-035495.

References

Alonso, M.E., Raimondo, M., 1990. Local Decomposition Algorithms. In: LNCS, vol. 508. Springer, pp. 208–221.
Becker, T., 1994. On Gröbner bases under specialization. Appl. Algebra Eng. Comm. Comput. 5, 1–8.
Becker, T., Weispfenning, V., 1991. The Chineses remainder problem, multivariate interpolation and Gröbner bases. In: Proceedings of ISSAC'91. ACM, pp. 64–69.
Becker, T., Weispfenning, V., 1993. Gröbner Bases: A Computational Approach to Commutative Algebra. Springer, New-York.
Caboara, M., Conte, P., Traverso, C., 1995. Yet Another Ideal Decomposition Algorithm. In: LNCS, vol. 1255. Springer, pp. 39–54.
Chen, C., Golubitsky, O., Lemaire, F., Moreno Maza, M., Pan, W., 2007. Comprehensive Triangular Decomposition. In: Proceedings of CASC'07. In: LNCS, vol. 4770. Springer Verlag, pp. 73–101.
Chen, C., Lemaire, F., Li, L., Moreno-Maza, M., Pan, W., Xie, Y., 2009. The ConstructibleSetTools and ParametricSystemTools modules for the RegularChains library in Maple. Preprint.
Coste, M., 2004. Classifying serial manipulators: Computer Algebra and geometric insight. Plenary talk (Personal communication). In: Proceedings of EACA-2004, pp. 323–323.
Dellière, S., (1999). Triangularisation de systèmes constructibles. Application à l'évaluation dynamique. Thèse Doctorale, Université de Limoges. Limoges, 1995.
Dolzmann, A., Seidl, A., Sturm, T., (2006). REDLOG software in REDUCE http://redlog.dolzmann.de/.
Duval, D., 1995. Évaluation dynamique et clôture algébrique en Axiom. J. Pure Appl. Algebra 99, 267–295.
Eisenbud, D., 1994. Commutative Algebra with a View Toward Algebraic Geometry. Springer, New York, (Corrected 3rd printing (1999)).
Eisenbud, D., Huneke, C., Vasconcelos, W., 1992. Direct methods for primary decomposition. Inventiones Math. 110, 207–235.
Emiris, I.Z., 1999. Computer algebra methods for studying and computing molecular conformations. Algorithmica 25, 372–402.
Faugère, J.C., 2002. A new efficient algorithm for computing Gröbner bases without reducing to zero (F_5). In: Proc. ISSAC'02. ACM, pp. 75–83.
Fortuna, E., Gianni, P., Trager, B., 2001. Degree reduction under specialization. J. Pure Appl. Algebra 164 (1–2), 153–164. (Proc of MEGA 2000).
Gao, X.S., Wang, D.K., 2003. Zero decomposition theorems for counting the number of solutions for parametric equation systems. In: Li, Ziming, Sit, William (Eds.), Proceedings of the 6th Asian Symposium on Computer Mathematics. In: Lecture Notes Series on Computing, vol. 10. World Scientific, pp. 129–144.
Gebauer, R., Möller, H.M., 1988. On an installation of Buchberger's algorithm. J. Symbolic Comput. 6, 275–286.
Gianni, P., 1987. Properties of Gröbner bases under specializations. In: Davenport, J.H. (Ed.), EUROCAL'87. In: LNCS, vol. 378. Springer, pp. 293–297.
Giovinni, A., Mora, T., Niesi, G., Robbiano, L., Traverso, C., 1991. "One sugar cube, please" OR Selection strategies in the Buchberger algorithm. In: Proc. ISSAC'91. ACM, pp. 49–54.
Giusti, M., Heintz, J., 1990. Algorithmes – disons rapides – pour la décomposition d'une variété algébrique en composantes irréductibles. Prog. Math. 94, 169–194.
Gómez-Díaz, T., 2000. Dynamic Constructible Closure. In: Proc. of Posso Workshop on Software, Paris, 2000, pp. 73–93.
González-López, M.J., Recio, T., 1993. The ROMIN inverse geometric model and the dynamic evaluation method. In: Cohen, A.M. (Ed.), Computer Algebra in Industry. Wiley & Sons, pp. 117–141.
González-López, M.J., González-Vega, L., Traverso, C., Zanoni, A., 2000. Gröbner bases specialization through Hilbert functions: the homogeneous case. SIGSAM BULL 34 (1), 1–8. (Issue 131).
González-Vega, L., Traverso, C., Zanoni, A., 2005. Hilbert stratification and parametric Gröbner bases. In: Proceedings of CASC-2005, pp. 220–235.
Gianni, P., Trager, B., Zacharias, G., 1988. Groebner bases and primary decomposition of polynomial ideals. J. Symbolic Comput. 6 (2–3), 149–167.

Van Hentenryck, P., McAllester, D., Kapur, D., 1997. Solving polynomial systems using a branch and prune approach. SIAM J. Numer. Anal. 34 (2), 797–827.

Heintz, J., Morgenstern, J., 1993. On the intrinsic complexity of elimination theory. J. Complexity 9, 471–498.

Inoue, S., Nagai, A., Sato, Y., 2007. On the computation of elimination ideals of boolean polynomial rings. In: Proceedings of ASCM, 2007, pp. 334–348.

Inoue, S., Sato, Y., 2007. On the parallel computation of comprehensive Gröbner systems. In: Proceedings of PASCO, 2007, pp. 99–101.

Kalkbrenner, M., 1997. On the stability of Gröbner bases under specializations. J. Symbolic Comput. 24 (1), 51–58.

Kapur, D., 1995. An approach for solving systems of parametric polynomial equations. In: Saraswat, Vijay, Van Hentenryck, Pascal (Eds.), Principles and Practices of Constraints Programming. MIT Press, pp. 217–244.

Manubens, M., 2008. Ph. Thesis "Parametric Polynomial System Discussion: Canonical Comprehensive Gröbner Systems", Universitat Politècnica de Catalunya, 2008.

Manubens, M., Montes, A., 2006. Improving DISPGB algorithm using the discriminant ideal. J. Symbolic Comput. 41, 1245–1263.

Manubens, M., Montes, A., 2009. Minimal canonical comprehensive Gröbner systems. J. Symbolic Comput. 44 (5), 463–478.

Montes, A., 1998. Algebraic solution of the load-flow problem for a 4-nodes electrical network. Math. Comp. Simul. 45, 163–174.

Montes, A., 2002. New algorithm for discussing Gröbner bases with parameters. J. Symbolic Comput. 33 (1–2), 183–208.

Mora, T., 2005. Solving Polynomial Equation Systems II: Macaulay's Paradigm and Gröbner Technology. Cambridge University Press, (Section 35.6-9).

Montes, A., Recio, T., 2007. Automatic discovery of geometry theorems using minimal canonical comprehensive Groebner systems. In: Proceedings of ADG 2006. In: LNAI, vol. 4869. Springer, pp. 113–138.

Moreno-Maza, M., (1997). Calculs de Pgcd au-dessus des Tours d'Éxtensions Simples et Résolution des Systèmes d'Équations Algebriques. Doctoral Thesis, Université Paris 6, 1997.

Nabeshima, K., 2005. A computation method for ACGB-V. In: Dolzmann, A., Seidl, A., Sturm, T. (Eds.), Proceedings of A3L 2005 (Conference in Honour of the 60th Bithday of V. Weispfenning). BOD Norderstedt, pp. 171–180.

Nabeshima, K., (2006). Comprehensive Groebner bases for modules. Short communication, ACA-2006, Varna.

O'Halloran, J., Schilmoeller, M., 2002. Gröbner bases for constructible sets. Comm. Algebra 30 (11), 5479–5483.

Pesch, M., (1994). Computing comprehesive Gröbner bases using MAS. User Manual, Sept. 1994.

Rychlik, M., 2000. Complexity and applications of parametric algorithms of computational algebraic geometry. In: del la Llave, R., Petzold, L., Lorenz, J. (Eds.), Dynamics of Algorithms. In: IMA Volumes in Mathematics and its Applications, vol. 118. Springer-Verlag, pp. 1–29.

Sato, Y., 2005. Stability of Gröbner basis and ACGB. In: Dolzmann, A., Seidl, A., Sturm, T. (Eds.), Proceedings of A3L 2005 (Conference in Honour of the 60th Birthday of V. Weispfenning). BOD Norderstedt, pp. 223–228.

Sato, Y., Suzuki, A., 2003. An alternative approach to comprehensive Gröbner bases. J. Symbolic Comput. 36 (3–4), 649–667.

Suzuki, A., Sato, Y., 2006. A simple algorithm to compute comprehensive Gröbner bases. In: Proceedings of ISSAC 2006. ACM, pp. 326–331.

Suzuki, A., Sato, Y., 2007. Implementation of CGS and CGB on Risa/Asir and other computer algebra systems using Suzuki–Sato algorithm. ACM Commun. Comput. Algebra 41 (3), http://kurt.cla.kobe-u.ac.jp/~sakira/CGBusingGB/.

Schönfeld, E., (1991). Parametrische Gröbnerbasen im Computeralgebrasystem ALDES/SAC-2. Dipl. Thesis, Universität Passau, Germany, May 1991.

Weispfenning, V., 1992. Comprehensive Gröbner bases. J. Symbolic Comput. 14 (1–1), 1–29.

Weispfenning, V., 2003. Canonical comprehensive Gröbner bases. J. Symbolic Comput. 36 (3–4), 669–683.

Wibmer, M., 2007. Gröbner bases for families of affine or projective schemes. J. Symbolic Comput. 42 (8), 803–834.

Yang, L., Hou, X., Xia, B., 2001. A complete algorithm for automated discovering of a class of inequality-type theorems. Sci. China, Ser. f 44 (6), 33–49.

Please cite this article in press as: Montes, A., Wibmer, M., Gröbner bases for polynomial systems with parameters. Journal of Symbolic Computation (2010), doi:10.1016/j.jsc.2010.06.017