An Implementation of the Characteristic Set Method in Maple*

Dongming Wang
Research Institute for Symbolic Computation
Johannes Kepler University, A-4040 Linz, Austria

This paper describes a complete implementation of Ritt-Wu's characteristic sets method in the Maple system. The implemented algorithms include those with variants for computing characteristic sets of (multivariate) polynomial sets, decomposing polynomial sets into ascending sets and irreducible ascending sets, decomposing algebraic varieties into irreducible components, factorizing polynomials over algebraic number fields and solving systems of polynomial equations. Some modification and generalization of the basic algorithms and implementation strategies are discussed. The timing statistics on a set of test problems is given.

1 Introduction and Notations

The method of characteristic sets was introduced by J. F. Ritt [5, 6] in the context of his work on differential algebra in the early 1930's and was revitalized and further developed by Wu Wen-tsün [11-13] through his recent work on mathematics-mechanization. In addition to be a powerful tool for Wu's general theory and method of mechanical theorem proving, the characteristic set method has proved efficient for solving a wide class of problems in geometry and algebra (see the series of work in [14] for example). It has been partially implemented by different research groups in China, USA and Austria [1, 3, 4, 14] for geometry theorem proving and solving other relevant problems. The author has learned that an implementation of this method in the Reduce system is ongoing at the University of Bath, England. However, to the best of our knowledge neither a complete implementation exists nor a partial implementation has been generally available in current symbolic and algebraic computation systems. The incompleteness of the existing implementations was mainly due to difficulties about polynomial factorization over successive algebraic extension fields (for which there was a lack of general and efficient procedures) and the determination of prime bases of ideals from their characteristic sets (for which no simple and practical method was available). We have overcome the difficulties through the discovery of a new method for polynomial factorization and the application of Gröbner bases for determining the prime bases.

In this paper we describe a complete, general-purpose implementation of the characteristic set method, considering mainly the zero structure of polynomial sets. This

---

*This work is supported in part by the Austrian Ministry of Science and Research under ESPRIT Basic Research Action 3125 (MEDLAR). A slightly revised version of this paper appeared in *Automated Practical Reasoning: Algebraic Approaches* (J. Pfalzgraf and D. Wang, eds.), Springer-Verlag, Wien-New York (1995), pp. 187–201.

implementation has been included in the Maple share library as a package under the name charsets[1] and can be considered as a practical basis for designing and implementing other related algorithms. The method of characteristic sets as well as its underlying theory has been well developed by Ritt and Wu. It provides rich contents for dealing with systems of (multivariate) polynomials (as well as differential polynomials). For the present implementation we essentially follow Wu's improved version of the method but use the algorithmic form described in [8] by taking most of the remarks there into account. The implemented algorithms include those for computing characteristic sets of polynomial sets, decomposing polynomial sets into ascending sets and irreducible ascending sets, decomposing algebraic varieties into irreducible components, factorizing polynomials over algebraic number fields and solving systems of polynomial equations. We have supplied several possible variants of these algorithms in order to make the package comprehensive and flexible.

The characteristic set method has an important application to mechanical theorem proving in geometries. On the basis of this package, the author has also developed a new and rather powerful geometry theorem prover GEOTHER which provides again a first complete implementation of Wu's general method. The prover has been treated as part of a geometry problem solver under development and is not included in this package. A detailed description of GEOTHER will be published elsewhere.

In the later sections we shall describe 15 user level functions available in our package, discuss the modification and generalization of some basic and utilized algorithms with our implementation strategies, and present a set of test results for all functions with variants. Before doing these, let us first explain our notations (which are similar to those used in [8]) in order to avoid confusion with notations used by other authors.

In the whole package all input polynomials are in parameters $u_1, \ldots, u_d$ and variables $x_1, \ldots, x_n$ with integer or rational coefficients. By a *constant* polynomial we mean one involving only the parameters. While the order of the variables is fixed, say

$$x_1 \prec x_2 \prec \cdots \prec x_n,$$

we call the variable with biggest index occurring in a non-constant polynomial F the *leading variable* of F, denoted as[2] $\mathrm{lvar}(F)$. The leading coefficient of a non-constant polynomial F with respect to $\mathrm{lvar}(F)$ is called the *initial* of F, denoted as $ini(F)$. A polynomial G is said to be *reduced* with respect to F if the degree of G in $\mathrm{lvar}(F)$ is less than the degree of F in $\mathrm{lvar}(F)$.

A finite set $\{A_1, A_2, \ldots, A_r\}$ of polynomials is called a *quasi-ascending set* or a *triangular form* if either $r = 1$ and $A_1 \neq 0$, or $r > 1$ and $\mathrm{lvar}(A_1) \prec \mathrm{lvar}(A_2) \prec \cdots \prec \mathrm{lvar}(A_r)$. A quasi-ascending set AS is called an *ascending set* if in the case $r > 1$, A_j is reduced with respect to A_i for each pair $j > i$. A quasi-ascending set AS is called a *weak ascending set* if in the case $r > 1$, the initial of A_j is reduced with respect to A_i for each pair $j > i$. A (weak, quasi-) ascending set is said to be *contradictory* if $r = 1$ and A_1 is a non-zero constant.

---

[1] The Maple code can be obtained via anonymous FTP at two sites: 129.132.101.33 (neptune) ETH Zurich, Switzerland and 129.97.140.58 (daisy) University of Waterloo, Canada. The printings of the help file and source code are also included in a technical report bearing the same title as this paper available as RISC-Linz Series no. 91-25.0.

[2] When we speak about $\mathrm{lvar}(F)$, it always implies that F is not a constant.

Let G be any non-zero polynomial and $\mathit{AS} = \{A_1, \ldots, A_r\}$ a non-contradictory (weak, quasi-) ascending set. One can pseudo-divide G successively by $A_r, \ldots, A_1$, considered as polynomials in their leading variables. The final remainder R is called the *pseudo-remainder* or simply the *remainder* of G with respect to AS.

Let PS be a finite, non-empty set of non-zero polynomials. The set of all non-zero remainders of the polynomials in PS with respect to a (weak, quasi-) ascending set AS is called the *remainder set* of PS with respect to AS. The set of all common zeros of the polynomials in PS is denoted by $\mathrm{Zero}(\mathit{PS})$. For any other non-zero polynomial G, we write $\mathrm{Zero}(\mathit{PS}/G)$ for $\mathrm{Zero}(\mathit{PS}) \setminus \mathrm{Zero}(G)$.

2 Description of User Functions

In this section we describe 15 user level functions which provide a great flexibility for using our package. There are two trivial functions iniset and remset, of which iniset computes the set of all distinct factors of initials of the polynomials in a non-contradictory (weak, quasi-) ascending set AS and remset computes the remainder set of a polynomial set PS with respect to AS. The other 13 non-trivial functions are given below.

2.1 charset and mcharset

A (weak, quasi-) ascending set CS is said to be a (*weak, quasi-*) *characteristic set* of a polynomial set PS if any polynomials in CS is a linear combination of the polynomials in PS with polynomial coefficients (i.e., CS is contained in the ideal generated by the polynomials in PS) and the remainder set of PS with respect to CS is empty. For a characteristic set CS of PS, we have therefore

$$\mathrm{Zero}(\mathit{CS}/J) \subset \mathrm{Zero}(\mathit{PS}) \subset \mathrm{Zero}(\mathit{CS}),$$

where J is the product of initials of the polynomials in CS. A (weak, quasi-) ascending set CS is said to be a *modified* (*weak, quasi-*) *characteristic set* of PS if

$$\mathrm{Zero}(\mathit{CS}/J) \subset \mathrm{Zero}(\mathit{PS}), \quad \mathrm{Zero}(\mathit{PS}/F) \subset \mathrm{Zero}(\mathit{CS}),$$

where J is the same as before and F is a non-zero polynomial.

The functions charset and mcharset compute respectively a (weak, quasi-) characteristic set and a modified (weak, quasi-) characteristic set of any polynomial set. For these two functions there is an option of 8 possible choices basset, wbasset, qbasset, charsetn, wcharsetn, qcharsetn, trisetc and triset for the so-called *medial sets*. These medial sets correspond respectively to those computed by the algorithms BasicSet, CharacteristicSetN, TriangularSetC and TriangularSet described in [7, 8]. If basset, charsetn or trisetc is chosen, then a characteristic set or a modified characteristic set is computed; if wbasset or wcharsetn is chosen, then a weak characteristic set or a modified weak characteristic set is computed; if qbasset, qcharsetn or triset is chosen, then a quasi-characteristic set or a modified quasi-characteristic set is computed. The default is charsetn.

2.2 charser, mcs, ecs and mecs

If a polynomial set PS and a sequence of (weak) ascending sets $CS_1, \ldots, CS_e$ are such that

$$\mathrm{Zero}(PS) = \bigcup_{i=1}^{e} \mathrm{Zero}(CS_i/J_i),$$

where each J_i is the product of initials of the polynomials in CS_i, then $\{CS_1, \ldots, CS_e\}$ is called a (*weak*) *characteristic series* of PS. If they are such that

$$\mathrm{Zero}(PS/G) = \bigcup_{i=1}^{e} \mathrm{Zero}(CS_i/F_i),$$

where each F_i is a polynomial having non-zero remainder with respect to CS_i, then $\{CS_1/F_1, \ldots, CS_e/F_e\}$ is called an *extended* (*weak*) *characteristic series* of PS/G.

Both the functions charser and mcs compute a (weak) characteristic series of PS, and so do both the functions ecs and mecs an extended (weak) characteristic series of PS/G. The only difference between charser and mcs and between ecs and mecs is: for the latter which is in general fast for large problems, some factors are examined and allowed to be removed during the internal computation of characteristic sets. Since we are unable to judge which function is better, both are kept in the package. For these functions there is an option of 5 medial sets basset, wbasset, charsetn, wcharsetn and trisetc, of which charsetn is again the default. A characteristic series or an extended characteristic series is computed if basset, charsetn or trisetc is chosen, and a weak characteristic series or a modified weak characteristic series is computed if any of the others is chosen.

2.3 triser and csolve

The function triser computes, from a polynomial set PS, a sequence of ascending sets, weak ascending sets or quasi-ascending sets $CS_1, \ldots, CS_e$ such that

$$\mathrm{Zero}(PS) = \bigcup_{i=1}^{e} \mathrm{Zero}(CS_i/J_i),$$

where each J_i is the product of initials of the polynomials in CS_i. It is designed mainly to prepare a sequence of triangular forms for solving the corresponding system of polynomial equations. The function csolve finds all solutions of a system of polynomial equations. It basically uses the function triser to prepare a sequence of triangular forms and then solves each triangular form by successive substitution, where the Maple function solve is used for the resolution of univariate polynomial equations.

2.4 qics, ics and eics

If all polynomials in the (weak) ascending sets of a characteristic series of PS are irreducible, then the (weak) characteristic series is said to be *quasi-irreducible*. If all ascending sets of a characteristic series or an extended characteristic series of PS are irreducible, then the characteristic series or extended characteristic series is said to be *irreducible*.

The functions qics, ics and eics compute respectively a quasi-irreducible (weak) characteristic series, an irreducible characteristic series, and an extended irreducible characteristic series of a polynomial set PS or PS/G. For qics there is an option of 5 medial sets basset, wbasset, charsetn, wcharsetn and trisetc too. A quasi-irreducible weak characteristic series is computed if wbasset or wcharsetn is chosen, and a quasi-irreducible characteristic series is computed if one of the others is chosen. For ics and eics, an option of 3 medial sets basset, charsetn and trisetc is allowed. In all three functions charsetn is again the default.

2.5 ivd

If a polynomial set PS and a sequence of polynomial sets $VS_1, \ldots, VS_t$ are such that

$$\mathrm{Zero}(PS) = \bigcup_{i=1}^{t} \mathrm{Zero}(VS_i),$$

and the algebraic variety defined by VS_i is irreducible for all i, then the above zero decomposition is called an *irreducible decomposition* of the algebraic variety defined by PS, and $VS_1, \ldots, VS_t$ are called the defining sets of irreducible components of the decomposition.

The function ivd computes a sequence of irredundant defining sets of the irreducible decomposition of the algebraic variety defined by a polynomial set PS. For this function, there is again an option of 3 medial sets basset, charsetn and trisetc with charsetn as default.

2.6 cfactor

Let $AS = \{A_1(u, y_1), A_2(u, y_1, y_2), \ldots, A_r(u, y_1, y_2, \ldots, y_r)\}$ (where u stands for $u_1, \ldots, u_d$) be an irreducible ascending set and $F = F(u, y_1, \ldots, y_r, y)$ any polynomial in $u, y_1, \ldots, y_r$, y with integer or rational coefficients. Consider F as a polynomial in y and suppose its leading coefficient has non-zero remainder with respect to AS. The function cfactor computes an irreducible factorization of F over the algebraic number field $\vec{Q}(u, y_1, \ldots, y_r)$, where $\vec{Q}$ denotes the rational number field, $u_1, \ldots, u_d$ are transcendental elements and $y_1, \ldots, y_r$ are algebraic elements with each y_i having minimal polynomial $A_i(u, y_1, \ldots, y_i)$.

3 Modifications and Strategies

The theory of characteristic sets developed by Ritt and Wu provides a constructive method for dealing with systems of polynomials (as well as differential polynomials). However, from a computational aspect many details have to be carefully taken into account for the sake of efficiency. These details are out of what Ritt was concerned. It was Wu who recognized the power of Ritt's method and considerably improved the method both in theory and in practice by bringing to it many new and important ideas. Through the design and implementation of our package we have made several further modifications and improvements and adopted a number of strategies. It is not possible to list all of them here but we want to mention a few as follows, of which some are given as remarks in [8].

3.1 Modification of the Pseudo-Division

The basic operation underlying all characteristic-set-based algorithms is the pseudo-division of two polynomials F and G with respect to a variable x. While dividing G by F, one gets a remainder formula of the form

$$I^s \cdot G = Q \cdot F + R,$$

where I is the leading coefficient of F in x. The integer s is determined to be as smallest so that the formula holds true according to Wu. Such a choice of the smallest s is important for reducing the degree and size of the remainder R. Now let us modify the above formula by replacing I^s with $I_1^{s_1} \cdots I_e^{s_e}$, where $I_1, \ldots, I_e$ are all the distinct irreducible factors of I, and choosing the smallest $s_1, \ldots, s_e$ so that the corresponding remainder formula holds still. For this modification the determination of R requires GCD (greatest common divisor) computation and thus takes more time at every individual step. However, we have observed that the modification can often avoid some redundant factors so that the subsequent computation profits, in particular, when the occurring polynomials become large. So in our package this modified pseudo-division is used.

3.2 Generalization of the Characteristic Set Algorithm

The characteristic set algorithm has several variants. Our implementation of this algorithm is based on a generalization described in [7]. We use the notion of *medial sets* of a polynomial set *PS* which are (weak, quasi-) ascending sets with rank not higher than that of the (weak, quasi-) basic set of *PS* and in which all polynomials are linear combinations of the polynomials in *PS* with polynomial coefficients. Thus, any (weak, quasi-) basic set itself is a special (weak, quasi-) medial set of the polynomial set. We proved that in Ritt's original algorithm, basic set can be replaced by any medial set. Besides basic set itself we have used in our implementation several other medial sets including those computed by the algorithms CharacteristicSetN, TriangularSetC, TriangularSet described in [7, 8] and by an algorithm designed for computing the so-called *normal* characteristic sets which are necessary for our factorization method to be mentioned below. With the notion of medial sets several variants of the characteristic set algorithm can be given in an uniform manner.

3.3 Removing Possible Factors of Intermediate Polynomials

During the computation of characteristic sets, there appear inevitably some redundant factors which are initials of other occurring polynomials. These factors should be removed in order to control the expansion of polynomial size. If one allows to remove factors, then the ascending sets computed by the characteristic set algorithm are no longer characteristic sets in Ritt's sense: they are what we call modified characteristic sets. Polynomials in a modified characteristic set are not necessarily elements of the ideal generated by the polynomials in the original set. If one considers only the zero structure of polynomial sets such as the case of solving polynomial equations, the modified characteristic sets are already well suited. If the removed factors are denoted by $F_1, \ldots, F_t$, then we have a zero relation of the form

$$\mathrm{Zero}(PS) = \mathrm{Zero}(CS/J) \cup \bigcup_{i=1}^{r} \mathrm{Zero}(PS_i) \cup \bigcup_{j=1}^{t} \mathrm{Zero}(QS_j),$$

where $J = I_1 \cdots I_r \cdot F_1 \cdots F_t$ and $PS_i = PS \cup \{I_i\}, QS_j = PS \cup \{F_j\}$.

In the functions mcharset, mcs, mecs, triser, qics, ics, eics and ivd, we all allow to remove possible factors. If the option of medial sets is triset or trisetc, we remove initials of the previous polynomials as factors from the newly produced polynomials in the computation of remainder sequence. Otherwise, we collect all the initials which have appeared in the previous computation and remove them as factors from the subsequently produced polynomials at certain stages. In all cases, the variables themselves as polynomials are examined and removed if possible. This process is of course time-consuming and does not seem to be recommendable for small problems. However, the removal of possible factors is very crucial for large problems. During our experiments we have seen that for many problems (e.g., Problem 4 as shown in Table 1) application of charset yields no result after a great amount of computing time, whereas mcharset gets the desired results easily. Since we are unable to predicate the computational cost for a given problem, we arrange to remove factors in most of our functions in order to avoid the trouble caused by large polynomials.

3.4 Polynomial Factorization

Polynomial factorization over both the rational number field and its algebraic extension fields is required in our implementation. As for factorization over the rationals, we use the Maple built-in function factor (either as a strategy for reducing the size of polynomials, or for computing the quasi-irreducible characteristic series, or as a subfunction of our procedure for factorizing polynomials over algebraic number fields).

To verify the reducibility of an ascending set, and if reducible, to decompose it into irreducible ones (which are needed in the functions ics, eics and ivd), we have to factorize polynomials over successive algebraic extensions of the rational number field which is considered as a difficult problem in general. Except Chou's implementation [1] which can factorize quadratic polynomials, all other implementations of the characteristic set method do not contain procedures for algebraic factorization. The general factorization algorithms are too complicated and not available in Maple. The author has implemented the factorization method presented in [2] which was considered suitable for our purpose. Experiments demonstrate that the method is feasible only for factorizing quadratic and cubic polynomials while degrees of the minimal polynomials are also not high. It is still too slow for polynomials of high degree. Recently, the author has found another method which can be used to factorize polynomials of rather high degree and reduces dramatically the difficulty of our factorization problem. This method has been implemented in combine with our early method as the function cfactor and used for the irreducible decomposition of ascending sets. The full details of our new method with experimental results are described in [10].

Since polynomial factorization is expensive in general, we have used some strategies for carrying out the factorization at an appropriate stage. For example, to verify the reducibility of a characteristic set we do the test soon after a medial set is computed in the option of `charsetn` and `trisetc`. Other strategies like factorizing initials and the removed factors in some cases are also very helpful for speeding up the computation.

3.5 Determining the Bases of Ideals from their Characteristic Sets

In our implemented algorithm for decomposing algebraic varieties into irreducible components, we first compute an irreducible characteristic series of the defining polynomial set and then determine the prime bases of ideals from the ascending sets in the series. For the latter an algorithm based on the Gröbner primbasissatz and Gröbner bases as described in [9] is implemented, where the Maple function `gbasis` in the `grobner` package is used for the computation of Gröbner bases.

3.6 Removing Redundant Branches of the Decomposition Tree

As argued in [9], the characteristic-set-based decomposition algorithms can be viewed as for computing a multi-branch decomposition tree. The number of branches of the tree can be hundred and thousand due to the recursive generation of initials. Some of the branches are completely redundant. Strategies must be used to avoid redundant computation in order to speed up the decomposition. In our implementation we have adopted various strategies for obtaining an equivalent but simpler tree with an aim at reducing the computing time and space. As most branches of the tree are produced from the recursive generation of initials, we observed that it is often profitable to decrease the depth and width of the decomposition tree by adjoining not the initials and the removed factors but the distinct irreducible factors of them. Even though this requires extensive polynomial factorization, the computation is not very expensive mainly because the initials and remove factors are relatively simple.

We cut off some redundant branches during the computation of various characteristic series according to a fact as follows: Let QS and QS' be two polynomial sets associated with two nodes of the decomposition tree of PS and $\mathit{QS} \subset \mathit{QS}'$. If the decomposition tree of QS is already computed, then the decomposition tree of QS' as a subtree of PS can be cut away. After the characteristic series has been produced, we remove some redundant ascending sets by another fact: For two ascending sets AS_1 and AS_2 of which the former is irreducible, if with respect to AS_1 the remainder set of AS_2 is empty and the remainder of J_2 is non-zero, then $\mathrm{Zero}(\mathit{AS}_1/J_1) \subset \mathrm{Zero}(\mathit{AS}_2/J_2)$, where J_i is the product of initials of the polynomials in AS_i for $i = 1, 2$, and thus AS_1 can be removed. For the function `ivd`, we also cut off some redundant branches by the affine dimension theorem in algebraic geometry: The dimension of an irredundant component of a variety is not less than $n - s$, where n is the number of variables and s the number of defining polynomials. Finally, all the redundant components are removed by using a lemma of Wu [12].

3.7 Optimization of Variable Ordering

The time for computing characteristic sets and thus all relevant decompositions depends heavily upon the choice of variable ordering. For example, if we order the variables in Problem 4 as $r \prec z \prec y \prec x$, then the characteristic set can be computed within one second. Therefore, the optimization of variable ordering must be considered for some applications such as the decomposition of algebraic varieties. In our implementation, if the variables are given as a set, a *heuristically* optimal variable ordering in the following sense is used.

Let X be a set of variables and PS a set of polynomials in X. For any variable $x \in X$ we define

$$\Omega(x, PS) = \max_{p \in PS} \deg(p, x), \quad \omega(x, PS) = \max\{1, \min_{p \in PS} \deg(p, x)\},$$

$$\Lambda(x, PS) = \text{number of elements in } \{p \in PS | \deg(p, x) = \Omega(x, PS)\},$$

$$\lambda(x, PS) = \text{number of elements in } \{p \in PS | \deg(p, x) = \omega(x, PS)\},$$

$$\Delta(x, PS) = \min_{p \in PS, \deg(p,x) = \omega(x,PS)} \text{Tdeg}(\text{lc}(p, x)),$$

$$\delta(x, PS) = \min_{p \in PS, \deg_x(p) = \omega(x,PS)} \text{Term}(\text{lc}(p, x)),$$

where $\deg(p, x)$ denotes the degree of p in x, $\text{lc}(p, x)$ the leading coefficient of p with respect to x, $\text{Tdeg}(q)$ the total degree of q and $\text{Term}(q)$ the number of actual terms in q.

Then, a partial order of the variables with respect to PS is introduced as follows:

1. If a variable $x \in X$ occurs in one and only one polynomial $p \in PS$, then x has order higher than all the others.
2. A variable x has order higher than the variable y if one of the following holds:
   (a) $\Omega(x, PS) < \Omega(y, PS)$;
   (b) $\Omega(x, PS) = \Omega(y, PS)$ but $\Lambda(x, PS) < \Lambda(y, PS)$;
   (c) $\Omega(x, PS) = \Omega(y, PS), \Lambda(x, PS) = \Lambda(y, PS)$ but $\omega(x, PS) < \omega(y, PS)$;
   (d) $\Omega(x, PS) = \Omega(y, PS), \Lambda(x, PS) = \Lambda(y, PS), \omega(x, PS) = \omega(y, PS)$ but $\lambda(x, PS) > \lambda(y, PS)$;
   (e) $\Omega(x, PS) = \Omega(y, PS), \Lambda(x, PS) = \Lambda(y, PS), \omega(x, PS) = \omega(y, PS), \lambda(x, PS) = \lambda(y, PS)$ but $\Delta(x, PS) < \Delta(y, PS)$;
   (f) $\Omega(x, PS) = \Omega(y, PS), \Lambda(x, PS) = \Lambda(y, PS), \omega(x, PS) = \omega(y, PS), \lambda(x, PS) = \lambda(y, PS)$, $\Delta(x, PS) = \Delta(y, PS)$ but $\delta(x, PS) < \delta(y, PS)$.

Under this partial order, the variables in Problem 4 can be optimally reordered as $z \prec x \prec y \prec r$.

4 Test Results and Remarks

Since all the functions plus their options in our package lead to many variants, we are unable to give a long list of test problems here due to the page restriction. We present some test results by taking only 3 problems with given variable ordering for each function. These problems were selected in a quite random way but so that they are representative and proper for testing most of the functions. For instance, the problems 1, 2 and 3 are all chosen so that polynomial factorization over algebraic number fields is involved. The experiments were made in Maple 4.3 running on an Apollo DN10000 under a UNIX operating system. All timings are given in CPU seconds without excluding the time for garbage collection. The timing statistics here aims at showing a rough magnitude about the computational cost of each function and does not necessarily reflect its overall behaviour. For the computation on very similar problems may lead to much different computing times while 3 test problems cannot say too much at all. In view of this we add below a few remarks on each function based on our extensive experiments with other problems.

Table 1. Timings for charset and mcharset

	Problem 2		Problem 3		Problem 4	
option	charset	mcharset	charset	mcharset	charset	mcharset
basset	>1800	161.633	14.100	15.650	>1800	540.400
charsetn	27.116	22.733	14.100	15.650	>1800	540.400
trisetc	21.933	34.850	4.350	6.717	>1800	>1800
wbasset	714.200	30.266	6.750	6.133	>1800	>1800
wcharsetn	23.084	7.600	3.466	3.766	>1800	1606.883
qbasset	1.684	2.250	2.967	2.984	>1800	105.600
qcharsetn	1.600	2.100	1.967	2.100	>1800	21.184
triset	1.383	2.350	2.150	4.333	>1800	29.817

Table 2. Timings for charser and mcs

	Problem 1		Problem 2		Problem 3	
option	charser	mcs	charser	mcs	charser	mcs
basset	1.383	1.550	>1800	1751.050	48.516	64.588
charsetn	1.017	1.283	59.616	70.966	18.850	35.734
trisetc	0.917	1.467	49.983	77.316	24.450	97.050
wbasset	3.067	1.617	1277.667	415.517	18.050	25.950
wcharsetn	3.766	1.467	46.967	45.250	33.133	21.517

Table 3. Timings for ecs and mecs

	Problem 1		Problem 2		Problem 3	
option	ecs	mecs	ecs	mecs	ecs	mecs
basset	1.716	1.883	>1800	816.567	48.900	53.750
charsetn	1.367	1.500	60.750	59.784	20.217	21.133
trisetc	0.900	1.067	48.283	65.484	27.217	26.800
wbasset	3.350	4.083	821.250	230.400	17.616	18.616
wcharsetn	3.834	3.416	44.500	33.317	32.017	14.067

Table 4. Timings for qics and eics

	Problem 1		Problem 2		Problem 3	
option	qics	eics	qics	eics	qics	eics
basset	3.583	11.367	359.867	384.783	188.233	239.150
charsetn	2.950	10.100	65.417	80.666	109.733	107.284
trisetc	4.417	12.717	61.850	99.583	173.800	158.000
wbasset	3.616		111.100		79.916	
wcharsetn	3.300		36.500		84.250	

Table 5. Timings for `ics` and `ivd`

	Problem 1		Problem 2		Problem 3	
option	ics	ivd	ics	ivd	ics	ivd
basset	10.533	12.616	369.100	381.683	279.733	352.000
charsetn	12.084	12.833	92.150	95.384	162.716	228.433
trisetc	12.950	14.983	61.334	75.650	274.150	398.583

Table 6. Timings for `triser`, `csolve` and `cfactor`

	triser	csolve
Problem 1	1.667	3.434
Problem 2	54.117	>1800
Problem 3	23.700	28.567

	cfactor
Problem 5	4.884
Problem 6	7.883
Problem 7	12.850

Remark 1. For (modified) characteristic sets, the computation in the quasi-sense usually is faster than that in the other senses, whereas the computation in the weak sense only sometimes is faster than that in the standard sense. However, the quasi-sense is theoretically weak. For example, in this sense we are unable to compute the characteristic series since the decomposition algorithm can no longer be guaranteed to terminate. Also, the irreducibility of a quasi-ascending set cannot be well defined. In the same sense, the computation is fast in general if the medial set `qcharsetn`, `wcharsetn`, `charsetn`, `triset` or `trisetc` is chosen. It is rather slow when the medial set `qbasset`, `wbasset` or `basset` is used. Here, the choice of `basset` results in Ritt's original version, the choice of (`w-`, `q-`) `charsetn` results in Wu's improved version of the algorithm, and the choice of `triset` and `trisetc` results in a version suggested by us.

Remark 2. `charset` sometimes is faster than `mcharset` for small problems, but it is slower for large ones. Consequently, `mcs` and `mecs` are respectively faster than `charser` and `ecs` for large problems. In general, the output of `mcs` and `mecs` is also more succinct than that of `charser` and `ecs`. The computing times for `charser` and `ecs`, for `mcs` and `mecs`, and for `ics` and `eics` may vary from each other as different strategies are used, but neither of them in a pair seems superior to the other. In `eics` an extended zero decomposition algorithm proposed by Wu [13] is implemented. It somewhat speeds up the computation in most cases but produces additional polynomials whose manipulation may cost time at other stages. Note that the implemented algorithms for the functions `mcs`, `qics` and `ics` have a similar structure and all the three examine and remove possible factors, while `qics` factorizes every polynomial in all ascending sets and `ics` decomposes all ascending sets into irreducible ones. The function `ics` is used as the main subfunction of `ivd`.

Remark 3. The function `triser` is designed for computing a hybrid sequence of ascending sets, weak ascending sets and quasi-ascending sets from a given polynomial set. The first few characteristic sets are computed in the quasi-sense for ease and the others are computed in the weak and standard senses in order to ensure the termination. This function is faster than the others for computing the characteristic series in some cases and may be used to prepare a sequence of triangular forms for polynomial equations solving. Actually, the function `csolve` is mainly based on `triser`.

Remark 4. During the implementation of our package we always keep in mind to attack large problems, where the term *large* is said with respect to the computational cost. Several strategies are used for this purpose and can reduce both the computing space and time for a number of large problems, but they may in turn increase the cost for small ones. Concerning the adoption of these strategies, we have difficulties to judge before the computation which kinds of problems are large. For this issue the author guesses that a careful analysis of the practical complexity for the used algorithms must be given first. Also, one variant of an algorithm may be fast for some problems but totally slow for others. We have tried to make choice from different algorithms and their variants by examining the sort of input according to our own experience. However, this works well only in some cases.

The characteristic sets and thus various related zero decompositions in general are not unique. To avoid redundant or unpleasant output expressions, we have also arranged to tidy up the output in case the postprocess is not expensive.

References

[1] Chou, S. C., *Mechanical Geometry Theorem Proving*, D. Reidel Publ. Co., Dord-recht-Boston-Lancaster-Tokyo (1988).

[2] Hu, S. and Wang, D. M., Fast Factorization of Polynomials over Rational Number Field or Its Extension Fields, *Kexue Tongbao* **31**, 150-156 (1986).

[3] Ko, H. P., ALGE-Prover II — A New Edition of ALGE-Prover, *Corp. Research & Development Tech. Info. Ser.*, General Electric, Schenectady, USA (1986).

[4] Kusche, K., Kutzler, B. and Mayr, H., Implementation of a Geometry Theorem Proving Package in Scratchpad II, *Proc. EUROCAL'87* (J. Davenport, ed.), Springer-Verlag, Heidelberg, 246-257 (1989).

[5] Ritt, J. F., *Differential Equations from the Algebraic Standpoint*, Amer. Math. Soc., New York (1932).

[6] Ritt, J. F., *Differential Algebra*, Amer. Math. Soc., New York (1950).

[7] Wang, D. M., A Generalization of Characteristic Sets Algorithm, *RISC-Linz Series* no. 89-51.0, Johannes Kepler Univ., Austria (1989).

[8] Wang, D. M., Characteristic Sets and Zero Structure of Polynomial Sets, *Lecture Notes*, RISC-LINZ, Johannes Kepler Univ., Austria (1989).

[9] Wang, D. M., Irreducible Decomposition of Algebraic Varieties via Characteristic Sets and Gröbner Bases, *Comput. Aided Geometric Design* **9**, 471-484 (1992).

[10] Wang, D. M., A Method for Factorizing Multivariate Polynomials over Successive Algebraic Extension Fields, *Preprint*, RISC-LINZ, Johannes Kepler Univ., Austria (1992).

[11] Wu, W. T., Basic Principles of Mechanical Theorem Proving in Elementary Geometries, *J. Sys. Sci. & Math. Scis.* **4**, 207-235 (1984); *J. Automated Reasoning* **2**, 221-252 (1986).

[12] Wu, W. T., *Basic Principles of Mechanical Theorem Proving in Geometries (Part on elementary geometries, in Chinese)*, Science Press, Beijing (1984).

[13] Wu, W. T., On Zeros of Algebraic Equations — An Application of Ritt Principle, *Kexue Tongbao* **31**, 1-5 (1986).

[14] *Mathematics-Mechanization Research Preprints*, No. 1-6, MM Research Center, Academia Sinica (1987-1991).

Appendix. Test Problems

The source of the following test problems: Problems 1-5 are taken from the papers by D. M. Wang, C. J. Butcher, W. T. Wu, M. Bronstein and P. S. Wang respectively. The polynomial F in Problem 6 is one that has to be factorized for computing the irreducible zero decompositions of PS in Problem 2. Problem 7 is a test example used by the author.

Problem 1. $\mathit{PS} = \{x_4^2+x_1x_4^2-x_2x_4-x_1x_2x_4+x_1x_2+3x_2, x_1x_4+x_3-x_1x_2, x_3x_4-2x_2^2-x_1x_2-1\}$ with variable ordering $x_1 \prec \cdots \prec x_4$.

Problem 2. $\mathit{PS} = \{p_1, \ldots, p_8\}$ with variable ordering $b \prec c_2 \prec c_3 \prec a \prec b_3 \prec b_2 \prec a_{32} \prec b_1$, where $p_1 = b_1 + b_2 + b_3 - a - b$, $p_2 = 2b_2c_2 + 2b_3c_3 - 1 - b - 2b^2 + 2ab$,
$p_3 = 3b_2c_2^2 + 3b_3c_3^2 - a - 3ab^2 + 4b + 3b^2 + 3b^3$,
$p_4 = 6b_3a_{32}c_2 - a - 3ab - 6ab^2 + 4b + 6b^2 + 6b^3$,
$p_5 = 4b_2c_2^3 + 4b_3c_3^3 - 1 - b - 10b^2 - 6b^3 - 4b^4 + 4ab + 4ab^3$,
$p_6 = 8b_3c_3a_{32}c_2 - 1 - 3b - 14b^2 - 12b^3 - 8b^4 + 4ab + 4ab^2 + 8ab^3$,
$p_7 = 12b_3a_{32}c_2^2 - 1 - b - 14b^2 - 18b^3 - 12b^4 + 8ab + 12ab^2 + 12ab^3$,
$p_8 = 1 + 7b + 26b^2 + 36b^3 + 24b^4 - 8ab - 24ab^2 - 24ab^3$.

Problem 3. $\mathit{PS} = \{y^2 - p_1, \frac{\partial p_2}{\partial x_1}, \frac{\partial p_2}{\partial x_2}, \frac{\partial p_2}{\partial x_3}, \frac{\partial p_2}{\partial x_4}, \frac{\partial p_2}{\partial x_5}, \frac{\partial p_2}{\partial x_6}, \frac{\partial p_2}{\partial \lambda_1}, \frac{\partial p_2}{\partial \lambda_2}, \frac{\partial p_2}{\partial \lambda_3}\}$ with variable ordering $x_1 \prec \cdots \prec x_6 \prec \lambda_1 \prec \lambda_2 \prec \lambda_3 \prec y$, where $p_1 = (x_4 + x_5)(x_5 + x_6)(x_6 + x_4)x_2^2x_1^2x_3^2$, $p_2 = p_1 + \lambda_1(x_2^2x_6 - 1) + \lambda_2(x_1^2x_4 - 1) + \lambda_3(x_3^2x_5 - 1)$.

Problem 4. $\mathit{PS} = \{x^2 + y^2 + z^2 - r^2, xy + z^2 - 1, xyz - x^2 - y^2 - z + 1\}$ with variable ordering $r \prec x \prec y \prec z$.

Problem 5. $\mathit{AS} = \{a^4 + a^3 + a^2 + a + 1\}$ and $F = 16x^4 + 8x^3 + 4x^2 + 2x + 1$.

Problem 6. $\mathit{AS} = \{-1 + b + 6b^2 + 12b^3\}$ and $F = 745092b - 252156 + 540900c + 21032664c^2b^2 + 2010720b^2+7117713c^2b-132367c^2+3076830c^3-7843500c^3b^2+2792322c^3b-3779244bc-10724400b^2c+ 21225240bc^5 + 26306208b^2c^5 + 8257464c^5 - 436536c^4 + 6094008b^2c^4 +594432bc^4$.

Problem 7. $\mathit{AS} = \{r^2 - 2 + z^2, -rz + y + 4y^2\}$ and $F = -370x^2y - 10x^3 + 60x^2z + 4xy - 24zy + 74rzy + 2rzx + 37rz - 37y + 12r^3 - 24r$ with variable ordering $z \prec y \prec x$.

Note. This paper was written in Winter 1990 when the first version of charsets was completed. It was accepted for publication in the Proceedings of DISCO '92 (Bath, April 13–15, 1992). However, the paper is not contained in the Proceedings (Springer LNCS **721**) because the editor did not receive its final camera-ready copy which the author sent in April 1992. After charsets's version 1.0 was included in the Maple share library for distribution, several improvements have been incorporated into the package. The up-to-date version of charsets is available via anonymous FTP on site 193.170.33.112 (ftp.risc.uni-linz.ac.at).