Algorithms for the universal decomposition algebra

Romain Lebreton
Laboratoire d’Informatique
École Polytechnique
Palaiseau - France

Éric Schost
Computer Science Department
University of Western Ontario
London - Canada

ALGO seminar, Rocquencourt

June 11, 2012

* This document has been written using the GNU TeXmacs text editor (see www.texmacs.org).
Let $k$ be a field of characteristic 0 or sufficiently large.

We fix $f = X^n + \sum_{i=1}^{n} (-1)^i f_i X^{n-i} \in k[X]$ separable of degree $n$.

We note $\alpha_1, \ldots, \alpha_n$ its roots.

Symmetric variety of roots of $f$

\[ \mathbb{V}_{I,k} = \{(\alpha_{\sigma(1)}, \ldots, \alpha_{\sigma(n)}) \mid \sigma \in \mathfrak{S}_n\} \leftrightarrow \quad I = \langle E_i(X_1, \ldots, X_n) - f_i \rangle_{i=1,\ldots,n} \subseteq k[X_1, \ldots, X_n] \]

The universal decomposition algebra is $A := k[X_1, \ldots, X_n]/I$, its degree is $\delta := n!$.

For all $P \in A$, let’s denote its characteristic polynomial

\[ \chi_{P,A}(T) := \prod_{\sigma \in \mathfrak{S}_n} (T - P(\alpha_{\sigma(1)}, \ldots, \alpha_{\sigma(n)})) \in k[T]. \]
State of the art: absolute resolvents

Absolute Lagrange’s resolvent:

\[ L_P(T) := \prod_{\sigma \in \mathcal{S}_n/\text{Stab } P} (T - P(\alpha_{\sigma(1)}, \ldots, \alpha_{\sigma(n)})) \in k[T]. \]

We have \( x_P = (L_P)^{\# \text{Stab } P} \).

Symbolic methods for the computation of absolute resolvents:

- by resultants [LAGRANGE], [SOICHER, ’81], [GIUSTI et al., ’88], [LEHOBEY, ’97]
- by symmetric functions [LAGRANGE], [VALIBOUZE, ’88], [CASPERSON, MCKAY, ’94];
- by Groebner bases [GIUSTI et al., ’88], [ARNAUDIÈS, VALIBOUZE, ’93];
- by invariants [BERWICK, ’29], [FOULKES, ’31].

\[\Rightarrow\] Little is known about complexity. Algorithm with at least quadratic complexity \( \Omega(\delta^2) \).
## State of the art: universal decomposition algebra

<table>
<thead>
<tr>
<th>Triangular representation</th>
<th>Univariate representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cauchy modules $C_i \in k[X_1, \ldots, X_i]$ for $1 \leq i \leq n.$</td>
<td>Minimal polynomial $Q$ of a primitive linear form $\Lambda$. Parametrizations $(S_i)_{1 \leq i \leq n}.$</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
A_1 &= k[X_1]/(C_1) \\
A_j &= k[X_1, \ldots, X_j]/(C_1, \ldots, C_j) \\
A &= A_n = k[X_1, \ldots, X_n]/(C_1, \ldots, C_n)
\end{align*}
\]

\[
\Lambda \simeq k[T]/(Q) \\
X_i \mapsto S_i(T) \\
\Lambda \mapsto T
\]

Cost of the representation of $A$:

- $O(\delta)$ by the recursive formula:
  
  \[
  C_{i+1} = \frac{(C_i(X_1, \ldots, X_i) - C_i(X_1, \ldots, X_{i+1}))(X_i - X_{i+1})}{X_i - X_{i+1}}
  \]
  
  with $C_1 := f(X_1)$.

- $\tilde{O}(\delta^3)$ by FGLM or RUR algorithm
  
  [Faugère et al., '93], [Rouillier, '99]

- $\tilde{O}(\delta^2)$ by geometric resolution
  
  [Giusti et al., '01], [Heintz et al., '00]

- $\tilde{O}(\delta^{1.69})$ by modular composition
  
  [Poteaux, Schost, '11]
## State of the art: universal decomposition algebra

<table>
<thead>
<tr>
<th>Triangular representation</th>
<th>Univariate representation</th>
</tr>
</thead>
</table>
| Cauchy modules $C_i \in k[X_1, \ldots, X_i]$ for $1 \leq i \leq n$ | Minimal polynomial $Q$ of a primitive linear form $\Lambda$. Parametrizations $(S_i)_{1 \leq i \leq n}$.

\[
A_1 = k[X_1]/(C_1) \\
\vdots \\
A_j = k[X_1, \ldots, X_j]/(C_1, \ldots, C_j) \\
\vdots \\
A = A_n = k[X_1, \ldots, X_n]/(C_1, \ldots, C_n)
\]

\[
A \simeq k[T]/(Q) \\
X_i \mapsto S_i(T) \\
\Lambda \mapsto T
\]

Cost of arithmetic operations in $A$:

- multiplication

  $\tilde{O}(\delta)$ [Bostan et al., 2011], not implemented, significant constant

  $\tilde{O}(\delta)$, simple and efficient

- division (when possible)

  no quasi-optimal algorithm

  $\tilde{O}(\delta)$, simple and efficient
Let $A_m = A \cap k[X_1, \ldots, X_m]$ and $\delta_m = \frac{n!}{(n-m)!}$ its degree.

**Theorem.** One can compute a primitive linear form $\Lambda \in A_m$ and the univariate representation $\Psi = (Q, S_1, \ldots, S_m) \in k[Z_m]^{m+1}$ with

\[
A_m = k[X_1, \ldots, X_m]/\mathcal{I} \simeq k[Z_m]/Q(Z_m)
\]

$X_i \mapsto S_i(Z_m)$

$\Lambda \mapsto Z_m$

with a Las Vegas algorithm of expected cost $O(n^{(\omega+1)/2} m M(\delta_m))$.

**Theorem.** For all $P \in A_m$, the characteristic polynomial $X_{P, A_m} \in k[T]$ costs

\[
O(n^{(\omega+1)/2} m M(\delta_m))
\]

arithmetic operations in $k$. 
Applications

- Computation of $X_P$
  $\mapsto$ Symbolic computation of absolute Lagrange’s resolvent in time $\tilde{O}(\delta_m)$;

- Computation of $A \simeq k[T]/Q(T)$
  $\mapsto$ Change of representation in time $\tilde{O}(\delta)$
  $\mapsto$ Division in $A$ in time $\tilde{O}(\delta)$
  $\mapsto$ Efficient algorithms for trace, minimal polynomial computations...
  $\mapsto$ Dynamic splitting field, [DELLA DORA et al., 1985]
  $\mapsto$ Effective invariant theory
Outline of the talk

Newton sums methods:
   i. Computation of $\mathcal{X}_\Lambda$ for a linear form $\Lambda \in A_m$
   ii. Change of representation: Up and Down
   iii. Univariate representation of $A_m$
   iv. Benchmarks

Resultant methods:
   i. Computation of $\mathcal{X}_P$ for any $P \in A_m$
   ii. Benchmarks
   iii. Generalizations
Newton sums

Definition. Let $g \in k[X]$ monic and $\beta_1, \ldots, \beta_n$ all its root in a suitable extension. Then the $i$-th Newton sum of $g$ is

$$S_i(g) := \sum_{\ell=1}^n (\beta_\ell)^i \in k.$$ 

The Newton representation of $g$ is $(S_i(g))_{0 \leq i \leq n}$.

Proposition. The conversion from and to the Newton representation can be done in time $O(M(n))$.

Lemma. Multiplication in the Newton representation:

$$S_i(fg) = S_i(f) + S_i(g).$$
Characteristic polynomial of linear forms

**Definition.** Let \( f, g \in k[T] \) such that \( f = \prod_{i=1}^{r} (T - \alpha_i) \), \( g = \prod_{j=1}^{s} (T - \beta_j) \) in \( \bar{k} \). Then

\[
\begin{align*}
f \oplus g := & \prod_{1 \leq i \leq r, 1 \leq j \leq s} (T - (\alpha_i + \beta_j)) \in k[T] \\
(\text{resp.}) \ f \otimes g := & \prod_{1 \leq i \leq r, 1 \leq j \leq s} (T - (\alpha_i \cdot \beta_j)) \in k[T]
\end{align*}
\]

**Proposition.** If \( \deg f, \deg g \leq n \), then \( f \otimes g \) and \( f \oplus g \) can be computed in time \( O(M(n^2)) \).

**Proof.** One has

\[
S_i(f \otimes g) = S_i(f) S_i(g)
\]

and

\[
\sum_{i \in \mathbb{N}} \frac{S_i(f \oplus g)}{i!} T^i = \left( \sum_{i \in \mathbb{N}} \frac{S_i(f)}{i!} T^i \right) \left( \sum_{i \in \mathbb{N}} \frac{S_i(g)}{i!} T^i \right).
\]

\[\square\]
Our goal: Let \( \Lambda = \lambda_1 X_1 + \cdots + \lambda_n X_n \in A_n \), compute

\[
\mathcal{X}_\Lambda(T) := \prod_{\sigma \in S_n} (T - (\lambda_1 \alpha_{\sigma(1)} + \cdots + \lambda_n \alpha_{\sigma(n)})).
\]

Examples:

- \( f \otimes (X - \lambda) = \prod_{i=1}^n (T - \lambda \alpha_i) = \mathcal{X}_{\lambda X_1, A_1} \)
- If \( R = \{\alpha_1, \ldots, \alpha_n\} \), then

\[
f \oplus f = \prod_{\alpha, \beta \in R} (T - (\alpha + \beta)) = \prod_{\alpha \neq \beta \in R} (T - (\alpha + \beta)) \prod_{\alpha \in R} (T - 2\alpha)
\]

\[
= \mathcal{X}_{X_1 + X_2, A_2} \cdot \mathcal{X}_{2X_1, A_2}.
\]
Characteristic polynomial of linear forms

Formula from [Caspersen, McKay, '94]:

\[ \chi_{X_1 + \cdots + X_m} = \prod_{h=1}^{m} \left( (\chi_{X_1 + \cdots + X_{m-h}}) \oplus (f \otimes (T - h)) \right)^{(-1)^{h+1}} \]

Proposition. (Generalization)

- One has

\[ \chi_{\Lambda_j, A_j}(T) = \frac{\chi_{\Lambda_{j-1}, A_{j-1}}(T) \oplus (f \otimes (T - \lambda_j))}{\prod_{i=1}^{j-1} \chi_{\Lambda_{j-1} + \lambda_j X_i, A_{j-1}}(T)}, \quad (1) \]

where \( \Lambda_j = \lambda_1 X_1 + \cdots + \lambda_j X_j \in A_j \).

- The associated recursive algorithm \textbf{NewtonSums} computes \( \chi_{\Lambda, A_m} \) in time \( O(2^n M(\delta_m)) \).

Advantages: Algorithm in the Newton representation, handle multiplicities, memoization

Drawback: Factor \( 2^n \)
Parametrizations

Relation characteristic polynomial / univariate representation:

- minimal polynomial: \( Q_j(T) = \mathcal{X}_{\Lambda_j}(T) \);
- parametrizations:

\[ \text{Lemma. Let } K := k[T_1, \ldots, T_n] \text{ and } \Lambda := T_1 X_1 + \cdots + T_n X_n \in K[X_1, \ldots, X_n]. \]

Then \( \mathcal{X}_\Lambda \in k[T, T_1, \ldots, T_n] \) and

\[ X_i = -\left( \frac{\partial \mathcal{X}_\Lambda}{\partial T_i} \right) / \left( \frac{\partial \mathcal{X}_\Lambda}{\partial T} \right) \] in \( \mathbb{A}_K \).

In practice, we use tangent numbers \( K := k[\varepsilon]/(\varepsilon^2) \) to compute derivatives:

- If \( \Lambda^\varepsilon := (\lambda_1 + \varepsilon) X_1 + \lambda_2 X_2 + \cdots + \lambda_n X_n \) then \( \mathcal{X}_{\Lambda^\varepsilon} = \mathcal{X}_{\Lambda} + \varepsilon \frac{\partial \mathcal{X}_\Lambda}{\partial T_i} \).
Newton sums methods:

i. Computation of $\mathcal{X}_\Lambda$ for a linear form $\Lambda \in \mathbb{A}_m$

ii. Change of representation: Up and Down

iii. Univariate representation of $\mathbb{A}_m$

iv. Benchmarks

Resultant methods:

i. Computation of $\mathcal{X}_P$ for any $P \in \mathbb{A}_m$

ii. Benchmarks

iii. Generalizations
**Lift-up and push-down**

**Goal:** Compute efficiently \( A_n = k[X_1, \ldots, X_n]/(C_1, \ldots, C_n) \longrightarrow k[Z_n]/(Q_n(Z_n)) \) and Down = Up\(^{-1}\).

**Elementary change of representation:**

\[
\begin{align*}
A_i[X_{i+1}]/(C_{i+1}) & \cong A_{i+1} \\
\uparrow & \\
\text{up}_i: & k[Z_i, X_{i+1}]/(Q_i(Z_i), C_{i+1}(Z_i, X_{i+1})) \longrightarrow k[Z_{i+1}]/(Q_{i+1})
\end{align*}
\]

**Example:**

\[
\text{Up: } A_3 = (k[Z_1]/C_1)[X_2, X_3]/(C_2, C_3) \longrightarrow_{\text{up}_1} (k[Z_2]/Q_2)[X_3]/(C_3) \longrightarrow_{\text{up}_2} k[Z_3]/(Q_3)
\]

with \( n = 3, \ Z_1 = X_1 \).
Lift-up and push-down

**Goal:** Compute efficiently

**Up:** $A_n = k[X_1, ..., X_n]/(C_1, ..., C_n) \longrightarrow k[Z_n]/(Q_n(Z_n))$ and $\text{Down} = \text{Up}^{-1}$.

**Elementary change of representation:**

$$k[Z_i, X_{i+1}]/(Q_i(Z_i), C_{i+1}(Z_i, X_{i+1})) \longrightarrow k[Z_{i+1}]/(Q_{i+1})$$

**up_i:**

| $Z_i$ | $\rightarrow$ | $Z_{i+1} - \lambda_{i+1} S_{i+1,i+1}(Z_{i+1})$ |
| $X_{i+1}$ | $\rightarrow$ | $S_{i+1,i+1}(Z_{i+1})$ |

**Algorithm up_i:**

**Input:** $P \in k[Z_i, X_{i+1}]$

**Output:** $\text{up}_i(P) \in k[Z_{i+1}]/(Q_{i+1})$

1. Compute $\tilde{P}(Z_i, X_{i+1}) = P(Z_i - \lambda_{i+1} X_{i+1}, X_{i+1}) \in (k[X_{i+1}]/f(X_{i+1})[Z_i] \quad \text{M}(n)\text{M}(\delta_i)$
2. Substitute $X_{i+1} \leftarrow S_{i+1,i+1}(Z_{i+1})$ in $\tilde{P}(Z_i, X_{i+1}) \quad \text{n M}(\delta_{i+1})$
Lift-up and push-down

Goal: Compute efficiently \( \text{Up: } \mathbb{A}_n \rightarrow k[Z_n]/(Q_n(Z_n)) \) and its converse map \( \text{Down} = \text{Up}^{-1} \).

Proposition.

1. Given \( Q_i, Q_{i+1} \) and \( S_{i+1,i+1} \), we can apply \( \text{up}_i \) in time \( \mathcal{O}(M(n) M(\delta_{i+1})) \).

2. Given \( (Q_i, S_{i,i})_{2 \leq i \leq n} \), we can apply \( \text{Up} \) in time \( \mathcal{O}(M(n) n M(\delta)) \).

Example:

\[
\text{Up: } \mathbb{A}_3 = (k[Z_1]/C_1)[X_2, X_3]/(C_2, C_3) \rightarrow_{\text{up}_1} (k[Z_2]/Q_2)[X_3]/(C_3) \rightarrow_{\text{up}_2} k[Z_3]/(Q_3)
\]

with \( n = 3, Z_1 = X_1 \).
Univariate representation of $A_m$

Algorithm - UnivRepNewtonSums

Input:
- $f \in k[T]$
- a primitive linear form $\Lambda := X_1 + \lambda_2 X_2 + \cdots + \lambda_n X_n$ of $A$

Output:
- a univariate representation $P_i = (Q_i, S_1, \ldots, S_n)$ of $A$.

Algorithm:
- Use NewtonSums to get for $2 \leq i \leq n$:
  - the minimal polynomials $Q_i$ $\sim O(2^n M(\delta))$
  - the last parametrizations $S_{i,i}$ of $A_i$ $\sim O(2^n M(\delta))$
- Get the other parametrizations: $S_i(Z_n) = Up(X_i)$ $\sim O(M(n) \cdot n \cdot M(\delta))$
Advantage:
- Good timings:

<table>
<thead>
<tr>
<th>n</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (sec)</td>
<td>Gröbner (F4) + FGLM</td>
<td>0.001</td>
<td>0.03</td>
<td>5.8</td>
<td>1500</td>
</tr>
<tr>
<td></td>
<td>NewtonSums</td>
<td>0.005</td>
<td>0.05</td>
<td>0.52</td>
<td>6.8</td>
</tr>
</tbody>
</table>

with Magma 2.17-1 over $k = \mathbb{F}_p$ with $p$ prime number of 28 bits.

Drawbacks:
- Complexity of UnivRepNewtonSums not quasi-optimal: $O(2^n M(\delta))$.
- NewtonSums does not compute $X_P$ for general $P \in \mathbb{A}$. 
Outline of the talk

Newton sums methods:
   i. Computation of $\mathcal{X}_\Lambda$ for a linear form $\Lambda \in \mathbb{A}_m$
   ii. Change of representation: Up and Down
   iii. Univariate representation of $\mathbb{A}_m$
   iv. Benchmarks

Resultant methods:
   i. Computation of $\mathcal{X}_P$ for any $P \in \mathbb{A}_m$
   ii. Benchmarks
   iii. Generalizations
Computation of $\mathcal{X}_P$ for any $P \in A$

Based on the resultant approach to compute resolvents:

- $R_n := T - P(X_1, ..., X_n) \in k[X_1, ..., X_n, T]$
- For $i = n - 1, ..., 0$, $R_i := \text{Res}_{X_{i+1}}(C_{i+1}, R_{i+1}) \in k[X_1, ..., X_i, T]$
- $\mathcal{X}_P = R_0 \in k[T]$.

Mathematically,

$$\text{Res}_{X_{i+1}} : A_i[T][X_{i+1}] \times A_i[T][X_{i+1}] \longrightarrow A_i[T].$$

Multiplication in $A_i$ in time $\tilde{O}(\delta_m)$ + resultant algorithm on general ring

$\leadsto$ Computation of $\mathcal{X}_P$ for any $P \in A_m$ in time $\tilde{O}(\delta_m)$

$\leadsto$ Not practical
Computation of $\chi_P$ for any $P \in A$

**Algorithm - ResultantCharPol**

**Input**: $P \in A$

**Output**: $\chi_{P,A}$

**Algorithm**:

1. $G_n := U_{p_{n-1}}(Y - P)$

2. for $i = n - 1 \ldots 1$ do
   
   $C_{i+1}^\prime := U_{p_i}(C_{i+1})$
   
   $G_i^\prime := \text{Res}_{X_{i+1}}(C_{i+1}^\prime, G_{i+1})$
   
   $G_i := \text{down}_{i-1}(G_i^\prime)$

3. return $G_i := \text{down}_{i-1}(G_i^\prime)$

**Cost of ResultantCharPol**: $O(n^{(\omega+1)/2} n M(\delta))$
Computation of $\mathcal{X}_P$ for any $P \in A$

**Algorithm - UnivRepResultant**

**Input :**
- $f \in k[T]$;
- a primitive linear form $\Lambda := X_1 + \lambda_2 X_2 + \ldots + \lambda_n X_n$ of $A$;

**Output :**
- a univariate representation $\Psi_i = (Q_i, S_1, \ldots, S_n)$ of $A$.

**Algorithm :**
- For $2 \leq i \leq n$, use $\text{ResultantCharPol}$ to get:
  - the minimal polynomials $Q_i$ \(\mathcal{O}(n^{(\omega+1)/2} M(\delta))\)
  - the last parametrizations $S_{i,i}$ of $A_i$ \(\mathcal{O}(n^{(\omega+1)/2} M(\delta))\)
- Get the other parametrizations: $S_i(Z_n) = Up(X_i)$ \(\mathcal{O}(M(n) n M(\delta_n))\)
Magma 2.17-1 over $k = \mathbb{F}_p$ with $p$ prime number of 28 bits.

### Characteristic polynomial for any $P \in \mathbb{A}$

<table>
<thead>
<tr>
<th>$n$</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (sec)</td>
<td>Traces in $k[Z]/Q(Z)$ [Shoup,'99]*</td>
<td>0.001</td>
<td>0.01</td>
<td>0.23</td>
<td>6.8</td>
</tr>
<tr>
<td></td>
<td>ResultantCharPol*</td>
<td>0.03</td>
<td>0.24</td>
<td>2.6</td>
<td>46</td>
</tr>
</tbody>
</table>

### Characteristic polynomial for $P \in \mathbb{A}$ linear

<table>
<thead>
<tr>
<th>$n$</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (sec)</td>
<td>NewtonSums</td>
<td>0.001</td>
<td>0.015</td>
<td>0.12</td>
<td>1.54</td>
</tr>
<tr>
<td></td>
<td>Traces in $k[Z]/Q(Z)$ [Shoup,'99]*</td>
<td><strong>0.001</strong></td>
<td><strong>0.005</strong></td>
<td><strong>0.10</strong></td>
<td>2.9</td>
</tr>
<tr>
<td></td>
<td>ResultantCharPol*</td>
<td>0.03</td>
<td>0.24</td>
<td>2.6</td>
<td>46</td>
</tr>
</tbody>
</table>

(*) Requires the precomputation of a univariate representation
**Benchmarks**

Magma 2.17-1 on one core of a Intel Xeon @2.27GHz, 74Gb of RAM over \( k = \mathbb{F}_p \) with \( p \) prime number of 28 bits.

<table>
<thead>
<tr>
<th>n</th>
<th>Up*</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>0.008</td>
<td>0.1</td>
<td>2</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>Down*</td>
<td>0.01</td>
<td>0.1</td>
<td>1.4</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>Univariate ( \times ) *</td>
<td>40 ( \mu s )</td>
<td>0.5 ms</td>
<td>0.006</td>
<td>0.06</td>
<td></td>
</tr>
<tr>
<td>Univariate ( \div ) *</td>
<td>0.002</td>
<td>0.03</td>
<td>0.29</td>
<td>4.5</td>
<td></td>
</tr>
<tr>
<td>Magma triangular ( \times )</td>
<td>0.003</td>
<td>0.09</td>
<td>4</td>
<td>170</td>
<td></td>
</tr>
<tr>
<td>Magma triangular ( \div )</td>
<td>0.1</td>
<td>28</td>
<td>&gt;30 min</td>
<td>&gt;6 h</td>
<td></td>
</tr>
</tbody>
</table>

(*) Requires the precomputation of a univariate representation
Newton sums methods:

i. Computation of $\lambda_{\Lambda}$ for a linear form $\Lambda \in A_m$

ii. Change of representation: Up and Down

iii. Univariate representation of $A_m$

iv. Benchmarks

Resultant methods:

i. Computation of $\lambda_{P}$ for any $P \in A_m$

ii. Benchmarks

iii. Generalizations
Generalizations

Generalization: Adapt the situation to $G \subseteq \mathcal{S}_n$.

Galoisian ideals:

\[ \mathcal{I}_G = \{ R \in k[X_1, \ldots, X_n] \mid \forall \sigma \in G, R(\alpha_{\sigma(1)}, \ldots, \alpha_{\sigma(n)}) = 0 \} \]

Remark. If $G = \text{Gal}(f)$, then $k[X_1, \ldots, X_n]/\mathcal{I}_G$ is a decomposition field of $f$.

Triangular sets of Galoisian ideals:

- [Orange, Renault, Valibouze, '03]
- [Lederer, '04]
- [Renault, Yokoyama, '06 '08]
- [Orange, Renault, Yokoyama, '09]
Generalizations

From triangular sets to univariate representation:

- Resultant approach is general
- **Up** and **Down** still in good complexity, due to $\forall i, f(X_i) = 0$

**Applications:**

- Computation of relative resolvents
  
  $$L_{P,G}(T) := \prod_{\sigma \in G/\text{Stab } P} (T - P(\alpha_{\sigma(1)}, \ldots, \alpha_{\sigma(n)})) \in k[T],$$

  where $P \in k[X_1, \ldots, X_n]^G$.

- Faster arithmetics in decomposition fields
Conclusion

Theoretical results:
- first quasi-linear algorithm for univariate representation in $A_m$;
- first quasi-linear algorithm for characteristic polynomial in $A_m$;
- Complexity improvement for the symbolic computation of absolute resolvents.

Practical results:
- MAGMA code;
- better timings for univariate representation of $A$, Up, Down;
- as a result, better timings for arithmetic operations in $A$, characteristic polynomial...
Thank you for your attention ;-)