# Classification of 3-dimensional integrable scalar discrete equations

Sergey P. Tsarev
Technical University Berlin, Germany
sptsarev@mail.ru

T. Wolf
Brock University, Ontario, Canada
twolf@brocku.ca

July 28, 2007

# Outline

# Outline

# The Setup

- **affine linearity of $Q$:**
  A 3d cube has $2^3 = 8$ corner values $f_{000}, f_{001}, \ldots, f_{111}$,

  $$\rightarrow Q = q_0 + q_1 f_{000} + q_2 f_{001} + \ldots + q_{255} f_{000} f_{001} \ldots f_{111}$$

  has $2^8 = 256$ terms with 256 undetermined coefficients $q_\alpha$.

# The Setup

- **affine linearity of $Q$:**
  A 3d cube has $2^3 = 8$ corner values $f_{000}, f_{001}, \ldots, f_{111}$,

  $$\rightarrow Q = q_0 + q_1 f_{000} + q_2 f_{001} + \ldots + q_{255} f_{000} f_{001} \ldots f_{111}$$

  has $2^8 = 256$ terms with 256 undetermined coefficients $q_\alpha$.
- **cubical symmetry of $Q$:**
  From 8 cases of cubical symmetry only 3 allow $Q \neq 0$ leaving 22, 13, 1 coefficients $q_\alpha$ undetermined.

# The Setup

- **affine linearity of $Q$:**
  A 3d cube has $2^3 = 8$ corner values $f_{000}, f_{001}, \ldots, f_{111}$,

  $$\rightarrow Q = q_0 + q_1 f_{000} + q_2 f_{001} + \ldots + q_{255} f_{000} f_{001} \ldots f_{111}$$

  has $2^8 = 256$ terms with 256 undetermined coefficients $q_\alpha$.
- **cubical symmetry of $Q$:**
  From 8 cases of cubical symmetry only 3 allow $Q \neq 0$ leaving 22, 13, 1 coefficients $q_\alpha$ undetermined.
- **4d consistency:**
  - Use 4 face relations to compute $f_{0111}, f_{1011}, f_{1101}, f_{1110}$ in terms of the 11 independent $f_{0000}, f_{0001}, f_{0010}, \ldots, f_{1010}, f_{1100}$.
  - Compute $f_{1111}$ 4 times from the remaining 4 face relations, require the equality of the 4 $f_{1111}$, giving 3 relations to be fulfilled identically in the 11 independent $f$'s.
  - Splitting wrt. the 11 independent $f$'s gives polynomial conditions for the unknowns $q_\alpha$.

# Outline

# Results

Case $(+ + +)$ : 5 solutions

Case $(- + +)$ : 3 solutions

Case $(- - -)$ : 1 solution

# A Solution with $(+++)$ Symmetry

$$
\begin{aligned}
Q \;=\;& q_{105}\left(f_{001}f_{010}f_{100}f_{111}+f_{000}f_{011}f_{101}f_{110}\right)+\\
& q_{107}\left(f_{001}f_{010}f_{100}f_{110}f_{111}+f_{001}f_{010}f_{100}f_{101}f_{111}+f_{001}f_{010}f_{011}f_{100}f_{111}+\right.\\
& \quad f_{000}f_{011}f_{101}f_{110}f_{111}+f_{000}f_{011}f_{100}f_{101}f_{110}+f_{000}f_{010}f_{011}f_{101}f_{110}+\\
& \quad \left.f_{000}f_{001}f_{011}f_{101}f_{110}+f_{000}f_{001}f_{010}f_{100}f_{111}\right)+\\
& \frac{q_{107}^2}{q_{105}}\left(f_{001}f_{010}f_{100}f_{101}f_{110}f_{111}+f_{001}f_{010}f_{011}f_{100}f_{110}f_{111}+\right.\\
& \quad f_{001}f_{010}f_{011}f_{100}f_{101}f_{111}+f_{000}f_{011}f_{100}f_{101}f_{110}f_{111}+\\
& \quad f_{000}f_{010}f_{011}f_{101}f_{110}f_{111}+f_{000}f_{010}f_{011}f_{100}f_{101}f_{110}+\\
& \quad f_{000}f_{001}f_{011}f_{101}f_{110}f_{111}+f_{000}f_{001}f_{011}f_{100}f_{101}f_{110}+\\
& \quad f_{000}f_{001}f_{010}f_{100}f_{110}f_{111}+f_{000}f_{001}f_{010}f_{100}f_{101}f_{111}+\\
& \quad \left.f_{000}f_{001}f_{010}f_{011}f_{101}f_{110}+f_{000}f_{001}f_{010}f_{011}f_{100}f_{111}\right)+\\
& \frac{q_{107}^3}{q_{105}^2}\left(f_{001}f_{010}f_{011}f_{100}f_{101}f_{110}f_{111}+f_{000}f_{010}f_{011}f_{100}f_{101}f_{110}f_{111}+\right.\\
& \quad f_{000}f_{001}f_{011}f_{100}f_{101}f_{110}f_{111}+f_{000}f_{001}f_{010}f_{100}f_{101}f_{110}f_{111}+\\
& \quad f_{000}f_{001}f_{010}f_{011}f_{101}f_{110}f_{111}+f_{000}f_{001}f_{010}f_{011}f_{100}f_{110}f_{111}+\\
& \quad \left.f_{000}f_{001}f_{010}f_{011}f_{100}f_{101}f_{111}+f_{000}f_{001}f_{010}f_{011}f_{100}f_{101}f_{110}\right)+\\
& 2\frac{q_{107}^4}{q_{105}^3}\left(f_{000}f_{001}f_{010}f_{011}f_{100}f_{101}f_{110}f_{111}\right)
\end{aligned}
$$

# Trivialization

Möbius transformations keep $0 = Q$ affine linear:

1. $f_{ijk} \mapsto \frac{q_{105}}{q_{107}} f_{ijk}$ and $Q \mapsto \frac{q_{107}^4}{q_{105}^5} Q$ (this eliminates the parametric $q_{105}$ and $q_{107}$)
2. $f_{ijk} \mapsto \frac{1}{f_{ijk}}$ (and removing the denominator in $Q$ afterwards)
3. $f_{ijk} \mapsto f_{ijk} - 1$.

This produces the simplified form
$0 = Q = f_{001} f_{010} f_{100} f_{111} + f_{000} f_{011} f_{101} f_{110}$ that can be linearized:
$\log(-\ldots) = \log(\ldots)$ showing that the solution is trivial.

# The Solution with $(- - -)$ Symmetry

**Main result**:
The following is the only *non-trivial* 3d affine linear face formula with cubical symmetry that is 4d consistent:

$$\begin{aligned} Q = (f_{100} - f_{001})(f_{010} - f_{111})(f_{101} - f_{110})(f_{011} - f_{001}) - \\ (f_{001} - f_{010})(f_{111} - f_{100})(f_{000} - f_{101})(f_{110} - f_{011}). \end{aligned} \tag{1}$$

This is the discrete Schwarzian bi-Kadomtsev-Petviashvili system (dBKP-system) — an integrable discrete system (Nimmo, Schief (1998) and Konopelchenko, Schief (2002)). It appears in many different contexts.

# Outline

# Size of Conditions

| dimension of face | $n$ | 2 | 3 | 4 |
|---|---|---|---|---|
| # of $f$-variables in face formula | $2^n$ | 4 | 8 | 16 |
| # of terms in face formula (= # of undetermined coefficients $q_{\mathcal{D}}$ in $Q_n$ ) | $2^{2^n}$ | 16 | 256 | 65536 |
| # of all $f$-variables in $(n+1)$-dim. hypercube | $2^{n+1}$ | 8 | 16 | 32 |
| # of indep. $f$-variables in $(n+1)$-dim. hypercube | $2^{n+1}-n-2$ | 4 | 11 | 26 |
| # of $n$-dim. faces in $(n+1)$-dim. hypercube | $2(n+1)$ | 6 | 8 | 10 |
| # of consistency conditions | $n$ | 2 | 3 | 4 |
| upper bound on the # of terms of each condition | $2^{\{2^{n+1}(n+1)-2n-1\}}$ | $5.2 \times 10^5$ | $1.4 \times 10^{17}$ | $2.8 \times 10^{45}$ |
| total degree of the $q_{\mathcal{D}}$ in each condition | $2n+2$ | 6 | 8 | 10 |
| upper bound estimate of the # of equations resulting from splitting each condition | $2n(2n+2)^{\left(2^{n+1}-n-3\right)}$ | 864 | $6.4 \times 10^9$ | $8.0 \times 10^{25}$ |
| estimated average # of terms in each equation | $\frac{2^{2^{n+1}(n+1)-2n-1}}{2n(2n+2)^{\left(2^{n+1}-n-3\right)}}$ | 606 | $2.2 \times 10^7$ | $3.5 \times 10^{19}$ |

Table: Size and number of consistency conditions

# Difficulties

1. Strictly speaking, in order to formulate even only the smallest subset of conditions one would have to formulate at least one consistency condition (by performing steps 1, 2 fully and 3 - 6 for at least two $x_k = 1$ face relations before splitting) i.e. to generate an expression with $2^{\{2^{n+1}(n+1)-2n-1\}}$ ($= 1.7 \times 10^{17}$ for $n = 3$) terms.

# Difficulties

1. Strictly speaking, in order to formulate even only the smallest subset of conditions one would have to formulate at least one consistency condition (by performing steps 1, 2 fully and 3 - 6 for at least two $x_k = 1$ face relations before splitting) i.e. to generate an expression with $2^{\{2^{n+1}(n+1)-2n-1\}}$ ($= 1.7 \times 10^{17}$ for $n = 3$) terms.
2. If one found a way around this hurdle then the resulting equations are of high degree $2n + 2$ with on average many terms.

# Difficulties

1. Strictly speaking, in order to formulate even only the smallest subset of conditions one would have to formulate at least one consistency condition (by performing steps 1, 2 fully and 3 - 6 for at least two $x_k = 1$ face relations before splitting) i.e. to generate an expression with $2^{\{2^{n+1}(n+1)-2n-1\}}$ ($= 1.7 \times 10^{17}$ for $n = 3$) terms.
2. If one found a way around this hurdle then the resulting equations are of high degree $2n + 2$ with on average many terms.
3. Even if one were able to generate 100,000's of equations and thus find shorter ones which one could solve for some unknowns in terms of others, one would face the problem that many cases and sub-sub-cases have to be investigated due to the high degree of the equations.

# Outline

# Simplifications I

Cubical symmetry is characterized by 3 $\pm$ in

$$Q \;=\; \pm\, Q|_{x\leftrightarrow y} \;=\; \pm\, Q|_{y\leftrightarrow -y} \;=\; \pm\, Q|_{z\leftrightarrow -z}.$$

From the 8 combinations of the 3 signs only 3 combinations allow not identically vanishing $Q$.

# Simplifications I

Cubical symmetry is characterized by 3 $\pm$ in

$$Q \;=\; \pm\, Q|_{x\leftrightarrow y} \;=\; \pm\, Q|_{y\leftrightarrow -y} \;=\; \pm\, Q|_{z\leftrightarrow -z}.$$

From the 8 combinations of the 3 signs only 3 combinations allow not identically vanishing $Q$.

For the hardest of these 3 symmetry types $(+\,+\,+)$ this leads to a reduction of the number of terms for each consistency condition from $10^{17}$ to about $10^{14}$ according to a study performed with the CA system FORM, although intermediate expressions are expected to have around $10^{15}$ terms.

# Cubical Symmetry

The nonempty symmetry classes of formulas $Q$ for face dimensions $2, 3, 4$ are:

| $n$ | types of symmetry,<br>number of parameters and terms | number of parameters<br>in $SL_2$-invariant subcases |
|---|---|---|
| 2 | $(+-)$: 1 param.; 4 terms<br>$(-+)$: 3 param.; 10 terms<br>$(++)$: 6 param.; 16 terms | 1 param.; 4 terms<br>none<br>1 param.; 6 terms |
| 3 | $(---)$: 1 param.; 24 terms<br>$(-++)$: 13 param.; 186 terms<br>$(+++)$: 22 param.; 256 terms | 1 param.; 24 terms<br>none<br>3 param.; 114 terms |
| 4 | $(----)$: 94 param.; 29208 terms<br>$(+---)$: 77 param.; 26112 terms<br>$(-+++)$: 349 param.; 60666 terms<br>$(++++)$: 402 param.; $2^{16}$ terms | 5 param.; 15480 terms<br>none<br>3 param.; 15809 terms<br>18 param.; 96314 terms |

Table: Symmetry classification of the face formulas w.r.t the complete symmetry group of the cube

# Simplifications II

**Probing**

There are 11 independent (splitting) variables
$f_\alpha : f_{0000}, f_{0001}, \ldots f_{1100}$ (with at most two 1's in the index).

- replace some ($:= z$) of the $f_\alpha$ by zero,
  - (+) smaller expressions, fewer unknowns $q_{\mathcal{D}}$

# Simplifications II

**Probing**

There are 11 independent (splitting) variables
$f_\alpha : f_{0000}, f_{0001}, \ldots f_{1100}$ (with at most two 1's in the index).

- replace some ($:= z$) of the $f_\alpha$ by zero,
  - (+) smaller expressions, fewer unknowns $q_{\mathcal{D}}$
  - (+) $\rightarrow$ tendency of triangularization because of 3d face formula
    $0 = Q = q_0 + q_1 f_{000} + \ldots + q_{255} f_{000} f_{001} f_{010} f_{011} f_{100} f_{101} f_{110} f_{111}$

# Simplifications II

**Probing**

There are 11 independent (splitting) variables
$f_\alpha : f_{0000}, f_{0001}, \ldots f_{1100}$ (with at most two 1's in the index).

- replace some ($:= z$) of the $f_\alpha$ by zero,
  - (+) smaller expressions, fewer unknowns $q_{\mathcal{D}}$
  - (+) $\rightarrow$ tendency of triangularization because of 3d face formula
    $0 = Q = q_0 + q_1 f_{000} + \ldots + q_{255} f_{000} f_{001} f_{010} f_{011} f_{100} f_{101} f_{110} f_{111}$
- replace some ($:= u$) of the remaining $f_\alpha$ by a random integer $\neq 0$
  - (+) smaller expressions
  - (–) fewer split variables

# Simplifications II

**Probing**

There are 11 independent (splitting) variables
$f_\alpha : f_{0000}, f_{0001}, \ldots f_{1100}$ (with at most two 1's in the index).

- replace some ($:= z$) of the $f_\alpha$ by zero,
  - (+) smaller expressions, fewer unknowns $q_{\mathcal{D}}$
  - (+) $\rightarrow$ tendency of triangularization because of 3d face formula
    $0 = Q = q_0 + q_1 f_{000} + \ldots + q_{255} f_{000} f_{001} f_{010} f_{011} f_{100} f_{101} f_{110} f_{111}$
- replace some ($:= u$) of the remaining $f_\alpha$ by a random integer $\neq 0$
  - (+) smaller expressions
  - (–) fewer split variables
- keep the remaining $s := 11 - z - u$ variables $f_\alpha$ symbolic.

# Probing in Action

- start with $z = 9, u = 0, s = 2,$

# Probing in Action

- start with $z = 9, u = 0, s = 2$,
- generate alg. conditions in steps 1-7, write them into a file

# Probing in Action

- start with $z = 9, u = 0, s = 2$,
- generate alg. conditions in steps 1-7, write them into a file
- start solution process, reading in equations from file when needed,

# Probing in Action

- start with $z = 9, u = 0, s = 2$,
- generate alg. conditions in steps 1-7, write them into a file
- start solution process, reading in equations from file when needed,
- generate a new file when all equations are read in,

# Probing in Action

- start with $z = 9, u = 0, s = 2$,
- generate alg. conditions in steps 1-7, write them into a file
- start solution process, reading in equations from file when needed,
- generate a new file when all equations are read in,
- repeat all until no new independent equations are generated

# Probing in Action

- start with $z = 9, u = 0, s = 2$,
- generate alg. conditions in steps 1-7, write them into a file
- start solution process, reading in equations from file when needed,
- generate a new file when all equations are read in,
- repeat all until no new independent equations are generated
- generalize parameters (change $u = 0$ to 1 and decrease $z$, or change $u = 1$ to 0 and increase $s$),

# Probing in Action

- start with $z = 9, u = 0, s = 2$,
- generate alg. conditions in steps 1-7, write them into a file
- start solution process, reading in equations from file when needed,
- generate a new file when all equations are read in,
- repeat all until no new independent equations are generated
- generalize parameters (change $u = 0$ to 1 and decrease $z$, or change $u = 1$ to 0 and increase $s$),
- continue generalization until all $q_{\mathcal{D}}$ appear and no new independent equations are generated,

# Probing in Action

- start with $z = 9, u = 0, s = 2$,
- generate alg. conditions in steps 1-7, write them into a file
- start solution process, reading in equations from file when needed,
- generate a new file when all equations are read in,
- repeat all until no new independent equations are generated
- generalize parameters (change $u = 0$ to 1 and decrease $z$, or change $u = 1$ to 0 and increase $s$),
- continue generalization until all $q_{\mathcal{D}}$ appear and no new independent equations are generated,
- solve the remaining equations in the solution process.

# Two Phases

1. Computing a solution by the above steps and generalizing to about $z = 1, u = 0, s = 10$ (limited by memory), so close to the general case $z = 0, u = 0, s = 11$.

# Two Phases

1. Computing a solution by the above steps and generalizing to about $z = 1, u = 0, s = 10$ (limited by memory), so close to the general case $z = 0, u = 0, s = 11$.
2. Giving a probabilistic proof by showing many times that all equations generated under $z = 0, u = 5, s = 6$ (which involve all $q_{\mathcal{D}}$) are solved by the obtained solution.

# Two Phases

1. Computing a solution by the above steps and generalizing to about $z = 1, u = 0, s = 10$ (limited by memory), so close to the general case $z = 0, u = 0, s = 11$.
2. Giving a probabilistic proof by showing many times that all equations generated under $z = 0, u = 5, s = 6$ (which involve all $q_{\mathcal{D}}$) are solved by the obtained solution.

   (Later by performing a brute force test using the computer algebra system FORM solutions are checked again rigorously and independently but it needs a special order of substitutions to be able to complete the test.)

# Negative Effects of Probing

(–) accidental vanishing of coefficients of $f_{11\ldots101\ldots1}$, $f_{11\ldots11}$ when trying to solve face relations for them,

# Negative Effects of Probing

(–) accidental vanishing of coefficients of $f_{11\ldots101\ldots1}$, $f_{11\ldots11}$ when trying to solve face relations for them,

(–) accidental factorization of some face relations,

# Negative Effects of Probing

(–) accidental vanishing of coefficients of $f_{11\ldots101\ldots1}$, $f_{11\ldots11}$ when trying to solve face relations for them,

(–) accidental factorization of some face relations,

(–) triangularization results in the need to generate many equations.

# Outline

# Triangularization

(+) Early equations involve only few unknowns $q_{\mathcal{D}}$ and can thus be solved / be simplified / be used to simplify others more easily.

# Triangularization

(+) Early equations involve only few unknowns $q_{\mathcal{D}}$ and can thus be solved / be simplified / be used to simplify others more easily.

(–) After generating the first $10^9$ equations (of the $6.4 \times 10^9$ equations) the obtained system is still not equivalent to the full system of conditions.

# Triangularization

(+) Early equations involve only few unknowns $q_{\mathcal{D}}$ and can thus be solved / be simplified / be used to simplify others more easily.

(–) After generating the first $10^9$ equations (of the $6.4 \times 10^9$ equations) the obtained system is still not equivalent to the full system of conditions.

**Solution:**
Use any relations of the form $q_i = q_i(q_j)$ known from the solution process in formulating consistency conditions.

(+) Millions of equations that are identically satisfied modulo the preliminary solution do not get formulated.

# Triangularization

(+) Early equations involve only few unknowns $q_{\mathcal{D}}$ and can thus be solved / be simplified / be used to simplify others more easily.

(–) After generating the first $10^9$ equations (of the $6.4 \times 10^9$ equations) the obtained system is still not equivalent to the full system of conditions.

**Solution:**

Use any relations of the form $q_i = q_i(q_j)$ known from the solution process in formulating consistency conditions.

(+) Millions of equations that are identically satisfied modulo the preliminary solution do not get formulated.

(+) Equations that get formulated are much shorter (involving only dozens or 100's of terms instead of 1000's or millions.

# Triangularization

(+) Early equations involve only few unknowns $q_{\mathcal{D}}$ and can thus be solved / be simplified / be used to simplify others more easily.

(–) After generating the first $10^9$ equations (of the $6.4 \times 10^9$ equations) the obtained system is still not equivalent to the full system of conditions.

**Solution:**
Use any relations of the form $q_i = q_i(q_j)$ known from the solution process in formulating consistency conditions.

(+) Millions of equations that are identically satisfied modulo the preliminary solution do not get formulated.

(+) Equations that get formulated are much shorter (involving only dozens or 100's of terms instead of 1000's or millions.

(–) As the computation splits into cases and $(\mathrm{sub}-)^{10-15}$cases the generated conditions are only valid in the corresponding case and its sub-cases $\longrightarrow$ very many files of generated equations (e.g. too many to do in unix: `rm *`).

# Solution

Have an automated (or if too difficult then interactive) automated solution process where solution steps alternate with reading new equations from a file and the generation of new files of equations.

# Solution

Have an automated (or if too difficult then interactive) automated solution process where solution steps alternate with reading new equations from a file and the generation of new files of equations.

(+) easy to do within CRACK with its flexible priority list. This needs just 2 more ‘modules’, one for generating a new file and one for reading from a file and it needs to find the right place for both modules within the priority list.

# Solution

Have an automated (or if too difficult then interactive) automated solution process where solution steps alternate with reading new equations from a file and the generation of new files of equations.

(+) easy to do within CRACK with its flexible priority list. This needs just 2 more ‘modules’, one for generating a new file and one for reading from a file and it needs to find the right place for both modules within the priority list.

(–) needs detailed programming if all should be done automatically for all sub-sub-cases.

# Accidental Factorization of Face Relations

We go back to the 2nd problem of ‘probing’.

For some face relation: $0 = A_k(f_\alpha) f_{1..101..1} + B_k(f_\alpha)$
it may be $P_k := GCD(A_k, B_k) \neq 1$ .

# Accidental Factorization of Face Relations

We go back to the 2nd problem of 'probing'.

For some face relation: $0 = A_k(f_\alpha) f_{1..101..1} + B_k(f_\alpha)$
it may be $P_k := GCD(A_k, B_k) \neq 1$ .
$\rightarrow$ substitutions $f_{1..101..1} = -B_k/A_k$ loose solutions which make
$P_k(q_\mathcal{D}, f_\alpha) = 0$

# Accidental Factorization of Face Relations

We go back to the 2nd problem of 'probing'.

For some face relation: $0 = A_k(f_\alpha) f_{1..101..1} + B_k(f_\alpha)$
it may be $P_k := GCD(A_k, B_k) \neq 1$ .
$\rightarrow$ substitutions $f_{1..101..1} = -B_k/A_k$ loose solutions which make $P_k(q_\mathcal{D}, f_\alpha) = 0$
$\rightarrow$ replace computed consistency conditions $0 = C_i(q_\mathcal{D}, f_\alpha)$ by conditions $0 = P_k C_i, \forall i$ and then split wrt. $f_\alpha$

# Accidental Factorization of Face Relations

We go back to the 2nd problem of 'probing'.

For some face relation: $0 = A_k(f_\alpha) f_{1..101..1} + B_k(f_\alpha)$
it may be $P_k := GCD(A_k, B_k) \neq 1$ .
$\rightarrow$ substitutions $f_{1..101..1} = -B_k/A_k$ loose solutions which make $P_k(q_\mathcal{D}, f_\alpha) = 0$
$\rightarrow$ replace computed consistency conditions $0 = C_i(q_\mathcal{D}, f_\alpha)$ by conditions $0 = P_k C_i, \forall i$ and then split wrt. $f_\alpha$
$\rightarrow$ better split first $P_k = 0 \rightarrow P_{kl} = 0$ and $C_i = 0 \rightarrow C_{ij} = 0$ and consider the system $0 = P_{kl} C_{ij}\ \forall l, i, j$.

# Accidental Factorization of Face Relations

We go back to the 2nd problem of 'probing'.

For some face relation: $0 = A_k(f_\alpha) f_{1..101..1} + B_k(f_\alpha)$
it may be $P_k := GCD(A_k, B_k) \neq 1$ .
$\rightarrow$ substitutions $f_{1..101..1} = -B_k/A_k$ loose solutions which make $P_k(q_\mathcal{D}, f_\alpha) = 0$
$\rightarrow$ replace computed consistency conditions $0 = C_i(q_\mathcal{D}, f_\alpha)$ by conditions $0 = P_k C_i, \forall i$ and then split wrt. $f_\alpha$
$\rightarrow$ better split first $P_k = 0 \rightarrow P_{kl} = 0$ and $C_i = 0 \rightarrow C_{ij} = 0$ and consider the system $0 = P_{kl} C_{ij}$ $\forall l, i, j$.
Repeat this for each factor of each GCD from each substitution, also the substitution of $f_{11...11}$.

# Consequence

(–) Equations are often factorizable leading to many cases and sub-cases.

# Consequence

(–) Equations are often factorizable leading to many cases and sub-cases.

(+) CRACK keeps track of inequalities
$\longrightarrow$ reduction of number of factors and thus equations,

# Consequence

(–) Equations are often factorizable leading to many cases and sub-cases.

(+) CRACK keeps track of inequalities
$\longrightarrow$ reduction of number of factors and thus equations,

(+) Also, this package is well suited to handle case distinctions automatically.

# Outline

# Summary

We solve a large problem (memory + time)

- by converting memory requirements into time requirements, and

# Summary

We solve a large problem (memory + time)

- by converting memory requirements into time requirements, and
- by solving the problem iteratively with successively increasing rigour and making use of gained information as soon as it becomes available.

# Summary

We solve a large problem (memory + time)

- by converting memory requirements into time requirements, and
- by solving the problem iteratively with successively increasing rigour and making use of gained information as soon as it becomes available.

Much of the development work is of universal nature and will be beneficial for future applications of CRACK.

# Summary

We solve a large problem (memory + time)

- by converting memory requirements into time requirements, and
- by solving the problem iteratively with successively increasing rigour and making use of gained information as soon as it becomes available.

Much of the development work is of universal nature and will be beneficial for future applications of CRACK.

One should not be afraid of large algebraic or even differential systems as long as they are heavily overdetermined and do not have too many solutions.