Symbolic Programming Examples
Mathematica vs FORM

Thomas Hahn, MPP

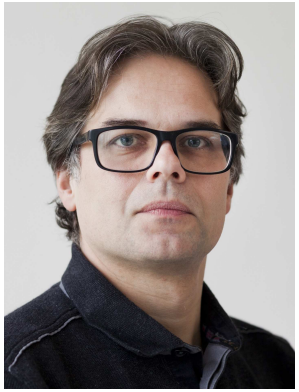
http://wwwth.mpp.mpg.de/members/hahn → **Lecture Material**

article in press, CA Rundbrief 63, Oct 2018

Computer Algebra als Grundvoraussetzung für das Gelingen der Vierten industriellen Revolution

C. Thiel, FHS St. Gallen Hochschule für Angewandte Wissenschaften, Institut für Informations- und Prozessmanagement IPM-FHS, Rosenbergstrasse 59, Postfach 9001 St. Gallen
C. Thiel, Fachhochschule Bielefeld, Campus Minden, Artilleriestr. 9, 32427 Minden

christian.thiel@fhsg.ch
christoph.thiel@fh-bielefeld.de



Kein Bild.

Einleitung

Industrie 4.0 betrifft uns alle und sorgt für einen tiefgreifenden Wandel in jedem Lebensbereich. Es entstehen für Innovatoren, Hersteller und Kunden neue bisher ungeahnte Chancen, aber auch neue Risiken. Die Beherrschung der wachsenden Komplexität ist bei vielen Aspekten nur mit Mitteln der Computer Algebra leistbar. Z.B. stellen weder Daten vnoch Vernetzung einen Mehrwert an sich dar. Stattdessen müssen viele Facetten der Computer Algebra ins Spiel kommen, um z.B. aus Big Data auch Smart Data zu machen. Dass dennoch der Mathematik, insbesondere auch der Computer Algebra, beim Thema Industrie 4.0 wenig Aufmerksamkeit zuteil

beläuft sich aktual auf ca. 8 Mrd. Euro jährlich. Produkte, die in Deutschland nachgeahmt werden, können oft als Hightech-Fälschungen betrachtet werden. Reverse Engineering (Hauptursache für 70 % der Fälschungen), Know-how-Verlust und Industriespionage (19 % aller Unternehmen) sind zu wichtigen Faktoren geworden. 82% der deutschen Unternehmen versuchen, sich mit rechtlichen Maßnahmen wie Patenten, Gebrauchsmustern und Geschmacksmustern vor Nachahmungen zu schützen. Leider ist dieser Ansatz reaktiv und greift nur dann, wenn der Schaden bereits eingetreten ist. Zudem geht weniger als die Hälfte der Unternehmen gegen Plagiate vor, sobald Eigentumsrechte verletzt werden. Aus diesen Gründen ist es besonders wichtig, präventive Schutzmaßnahmen (technisch und organisatorisch) zu

Mathematica in a Nutshell

Map applies a function to all elements of a list:

```
Map[f, {a, b, c}] ☞ {f[a], f[b], f[c]}
f /@ {a, b, c} ☞ {f[a], f[b], f[c]}   — short form
```

Apply exchanges the head of a list:

```
Apply[Plus, {a, b, c}] ☞ a + b + c
Plus @@ {a, b, c} ☞ a + b + c   — short form
```

Pure Functions are a concept from formal logic. A pure function is defined 'on the fly':

```
(# + 1)& /@ {4, 8} ☞ {5, 9}
```

The # (same as #1) represents the first argument, and the & defines everything to its left as the pure function.

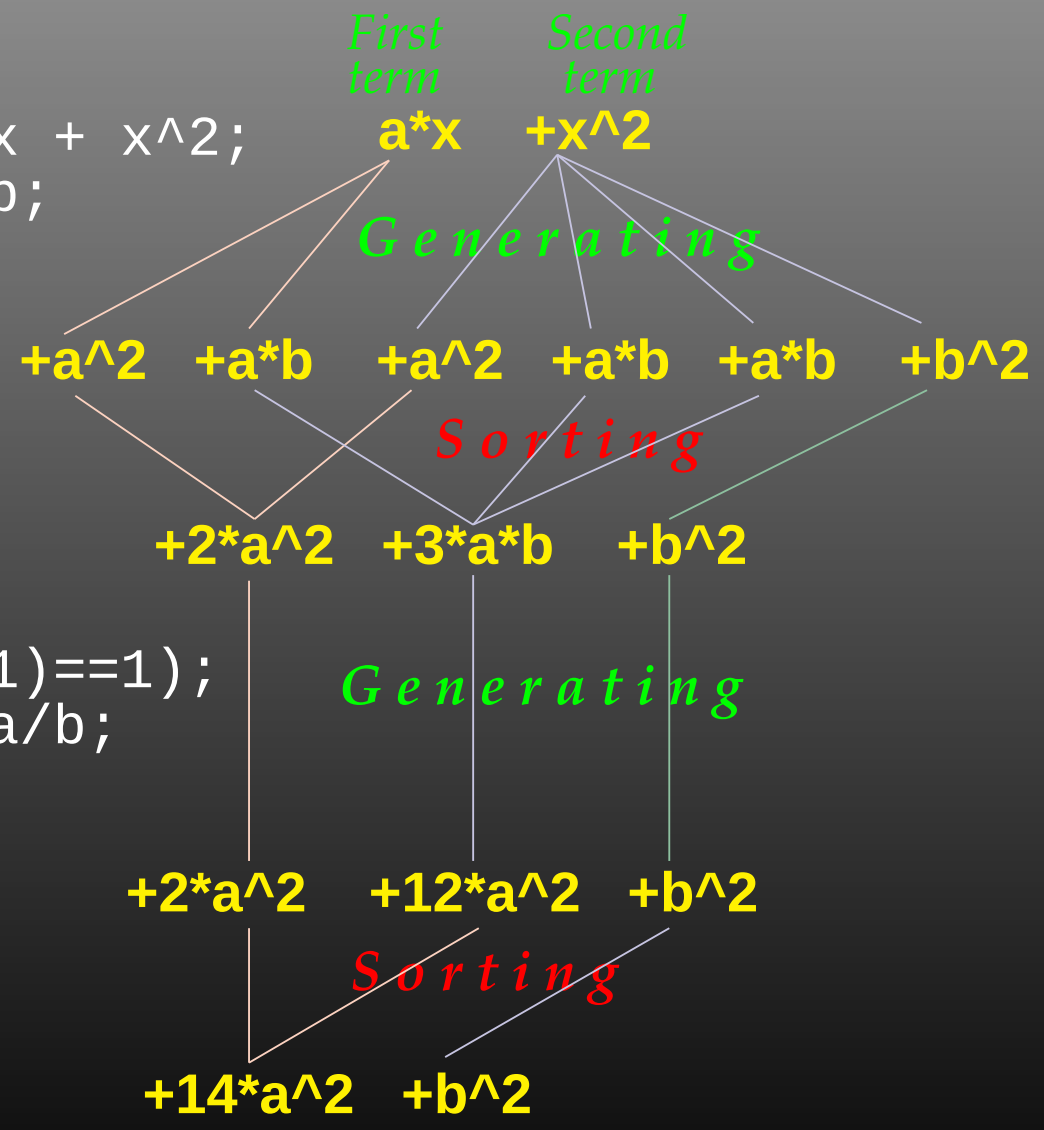
FORM in a Nutshell

First module

```
l expr = a*x + x^2;
id x = a + b;
.sort
```

Second module

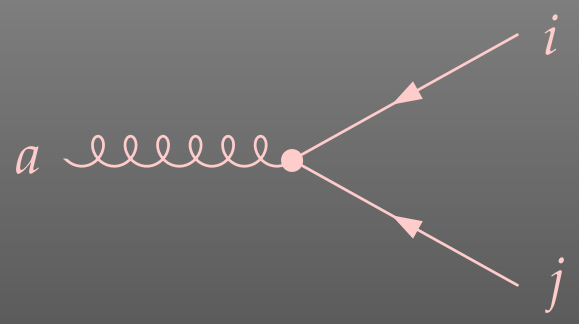
```
if(count(b,1)==1);
multiply 4*a/b;
endif;
print;
.end
```



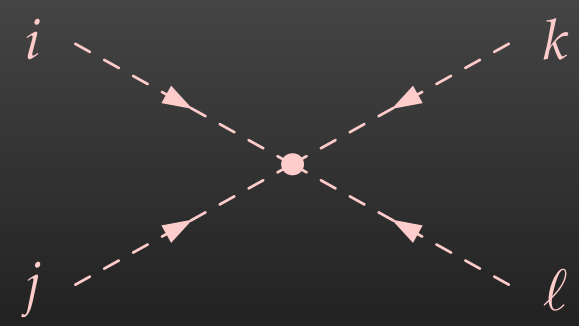
Color Structures

In Feynman diagrams four types of Color structures appear:

Natural Representation

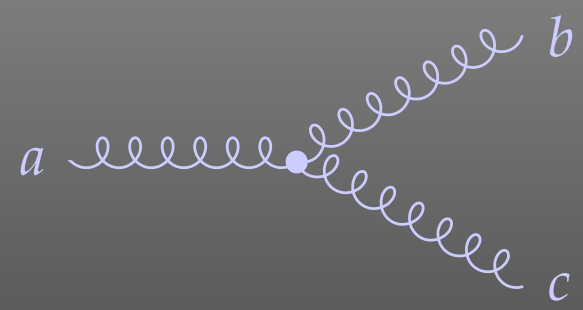


$\sim T^a_{ij} = \texttt{SUNT}[a,i,j]$

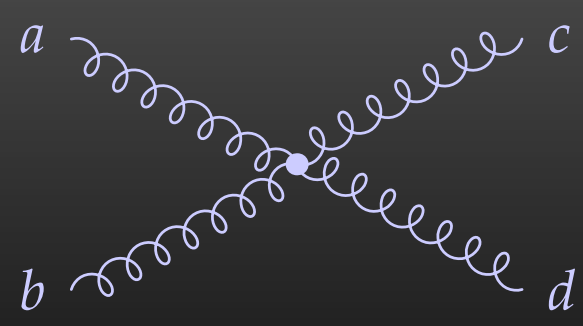


$\sim T^a_{ij} T^a_{k\ell} = \texttt{SUNTSum}[i,j,k,\ell]$

Adjoint Representation



$\sim f^{abc} = \texttt{SUNF}[a,b,c]$



$\sim f^{abx} f^{xcd} = \texttt{SUNF}[a,b,c,d]$

Unified Notation

The SUNF's can be converted to SUNT's via

$$f^{abc} = 2\mathrm{i}\left[\mathrm{Tr}(T^c T^b T^a) - \mathrm{Tr}(T^a T^b T^c)\right].$$

We can now represent all color objects by just SUNT:

- SUNT[i, j] $= \delta_{ij}$
- SUNT[a, b, . . . , i, j] $= (T^a T^b \cdots)_{ij}$
- SUNT[a, b, . . . , 0, 0] $= \mathrm{Tr}(T^a T^b \cdots)$

This notation again avoids unnecessary dummy indices. (Mainly namespace problem.)

For purposes such as the "large-N_c limit" people like to use SU(N) rather than an explicit SU(3).

Fierz Identities

The Fierz Identities relate expressions with different orderings of external particles. The Fierz identities essentially express completeness of the underlying matrix space.

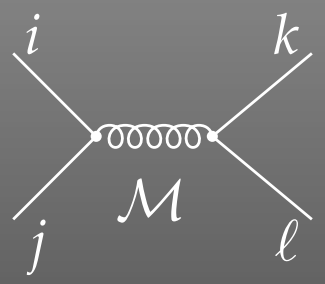
They were originally found by Markus Fierz in the context of Dirac spinors, but can be generalized to any finite-dimensional matrix space [hep-ph/0412245].

For SU(N) (color) reordering, we need

$$T^a_{ij} T^a_{k\ell} = \frac{1}{2} \left(\delta_{i\ell} \delta_{kj} - \frac{1}{N} \delta_{ij} \delta_{k\ell} \right) .$$

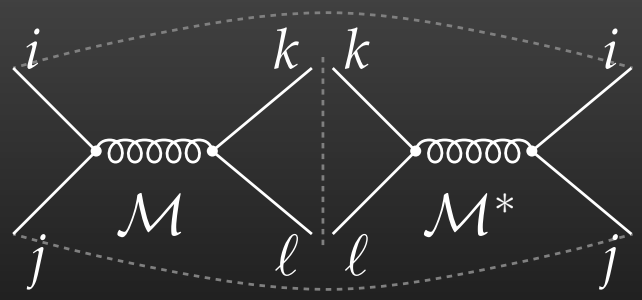
Cvitanovich Algorithm

For an **Amplitude:**



- convert all color structures to (generalized) `SUNT` objects,
- simplify: apply Fierz identity on all internal gluon lines,
- expect `SUNT` with indices of external particles to remain.

For a **Squared Amplitude:**



- use the Fierz identity to get rid of all `SUNT` objects,
- expect `SUNT` to vanish, color factors (numbers) only.

For "hand" calculations, a pictorial version of this algorithm exists in the literature.

Color Simplify in FORM

```
* introduce dummy indices for the traces
repeat;
  once SUNT(?a, 0, 0) = SUNT(?a, DUMMY, DUMMY);
  sum DUMMY;
endrepeat;

* take apart SUNTs with more than one T
repeat;
  once SUNT(?a, [a]?, [b]?, [i]?, [j]?) =
    SUNT(?a, [a], [i], DUMMY) * SUNT([b], DUMMY, [j]);
  sum DUMMY;
endrepeat;

* apply the Fierz identity
id SUNT([a]?, [i]?, [j]?) * SUNT([a]?, [k]?, [l]?) =
  1/2 * SUNT([i], [l]) * SUNT([j], [k]) -
  1/2/(`SUNN') * SUNT([i], [j]) * SUNT([k], [l]);
```

Translation to Color-Chain Notation

In color-chain notation we can distinguish two cases:

a) Contraction of different chains:

$$\langle A| T^a |B\rangle \langle C| T^a |D\rangle = \frac{1}{2}\left(\langle A|D\rangle \langle C|B\rangle - \frac{1}{N}\langle A|B\rangle \langle C|D\rangle\right),$$

b) Contraction on the same chain:

$$\langle A| T^a |B| T^a |C\rangle = \frac{1}{2}\left(\langle A|C\rangle \operatorname{Tr} B - \frac{1}{N}\langle A| B |C\rangle\right).$$

Color Simplify in Mathematica

```
   (* same-chain version *)
sunT[t1___, a_Symbol, t2___, a_, t3___, i_, j_] :=
  (sunT[t1, t3, i, j] sunTrace[t2] -
    sunT[t1, t2, t3, i, j]/SUNN)/2

   (* different-chain version *)
sunT[t1___, a_Symbol, t2___, i_, j_] *
sunT[t3___, a_, t4___, k_, l_] ^:=
  (sunT[t1, t4, i, l] sunT[t3, t2, k, j] -
    sunT[t1, t2, i, j] sunT[t3, t4, k, l]/SUNN)/2

   (* introduce dummy indices for the traces *)
sunTrace[a__] := sunT[a, #, #]&[ Unique["col"] ]
```

Fermion Trace

Leaving apart problems due to γ_5 in d dimensions, we have as the main algorithm for the 4d case:

$$\begin{aligned}\mathrm{Tr}\,\gamma_\mu\gamma_\nu\gamma_\rho\gamma_\sigma\cdots = &+ g_{\mu\nu}\,\mathrm{Tr}\,\gamma_\rho\gamma_\sigma\cdots \\ &- g_{\mu\rho}\,\mathrm{Tr}\,\gamma_\nu\gamma_\sigma\cdots \\ &+ g_{\mu\sigma}\,\mathrm{Tr}\,\gamma_\nu\gamma_\rho\cdots\end{aligned}$$

This algorithm is recursive in nature, and we are ultimately left with

$$\mathrm{Tr}\,\mathbb{1} = 4\,.$$

(Note that this 4 is not the space-time dimension, but the dimension of spinor space.)

Fermion Trace in Mathematica

```
Trace4[mu_, g__] :=
Block[ {Trace4, s = -1},
  Plus@@ MapIndexed[
    ((s = -s) Pair[mu, #1] Drop[Trace4[g], #2])&,
    {g} ]
]

Trace4[] = 4
```

Abbreviationing

One of the most powerful tricks to both reduce the size of an expression and reveal its structure is to substitute subexpressions by new variables.

The essential function here is `Unique` with which new symbols are introduced. For example,

```
Unique["test"]
```

generates e.g. the symbol `test1`, which is guaranteed not to be in use so far.

The `Module` function which implements lexical scoping in fact uses `Unique` to rename the symbols internally because Mathematica can really do dynamical scoping only.

Abbreviationing in Mathematica

```
$AbbrPrefix = "c"

abbr[expr_] := abbr[expr] = Unique[$AbbrPrefix]

   (* abbreviate function *)
Structure[expr_, x_] := Collect[expr, x, abbr]

   (* get list of abbreviations *)
AbbrList[] := Cases[DownValues[abbr],
  _[_[_[f_]], s_Symbol] -> s -> f]

   (* restore full expression *)
Restore[expr_] := expr /. AbbrList[]
```

Abbreviationing in FORM

```
* collect w.r.t. some function

b Den;
.sort
collect acc;

* introduce abbreviations for prefactors

toPolynomial onlyfunctions acc;
.sort

* print abbreviations & abbreviated expr

#write "%X"
print +s;
```