ISSN 1547-4771, Physics of Particles and Nuclei Letters, 2009, Vol. 6, No. 7, pp. 530–534. © Pleiades Publishing, Ltd., 2009.

An Application of Computer Algebra System Cadabra to Scientific Problems of Physics[1]

L. A. Sevastianov, D. S. Kulyabov, and M. G. Kokotchikova

Peoples' Friendship University of Russia

e-mail: sevast@sci.pfu.edu.ru, dharma@sci.pfu.edu.ru, mgkokotchikova@gmail.com

Abstract—In this article we present two examples solved in a new problem-oriented computer algebra system Cadabra. Solution of the same examples in widespread universal computer algebra system Maple turn out to be more difficult.

PACS number: 03.67.Lx; 01.30.Cc; 03.67.-a

DOI: 10.1134/S1547477109070073

1. INTRODUCTION

The goal of all computer algebra systems (CAS) is to help researchers in manipulations with formulas in physics, mechanics, algebra, analysis problems. Unhandiness of mathematical operations force to slow down research process and so automatization of formula manipulation is important. Automatization can help to reduce research time. Furthermore, there are problems which can't be solved without automatization of formula manipulation.

CAS is a software program that facilitates symbolic mathematics. The core functionality of a CAS is manipulation of mathematical expressions in symbolic form.

Actually, there are about 30 CAS (http://en.wikipedia.org/wiki/Comparison_of_computer_algebra_systems) and from this systems we have chosen .5 following systems: Maple—http://www.maplesoft.com/, Mathematica—http://www.wolfram.com/, Axiom—http://wiki.axiom-developer.org/, Maxima—http://maxima.sourceforge.net/ and Cadabra—http://www.aei.mpg. de/@~peekas/cadabra/.

We have classified them by area of application. Maple, Mathematica, Axiom, Maxima are universal systems. Cadabra is a problem-oriented system. It means that in a universal system there are many general methods, but problem-oriented system was made for particular tasks.

Kasper Peeters, the high-energy physicists, is a creator of Cadabra. He works now at the Institute for Theoretical Physics of Utrecht University in The Netherlands and his area of research is located on the overlap between (quantum) field theory, gravity and string theory. He created Cadabra in 2007 for research in field theory and general relativity which are not available in other systems.

[1] The article is published in the original.

2. APPLICATION OF CADABRA

CAS Cadabra has called our attention because its particular tasks are in following areas:

—Field theory,

—Quantum mechanics,

—Quantum field theory,

—Gravity,

—Supergravity,

—General relativity.

Cadabra is a new CAS designed specifically for the solution of problems encountered in field theory. It has extensive functionality for tensor polynomial simplification taking care of Bianchi and Schouten identities, for fermions and anti-commuting variables, Clifford algebras and Fierz transformations, implicit coordinate dependence, multiple index types and many other field theory related concepts. The input format is a subset of $\mathrm{T_EX}$ and thus easy to learn.

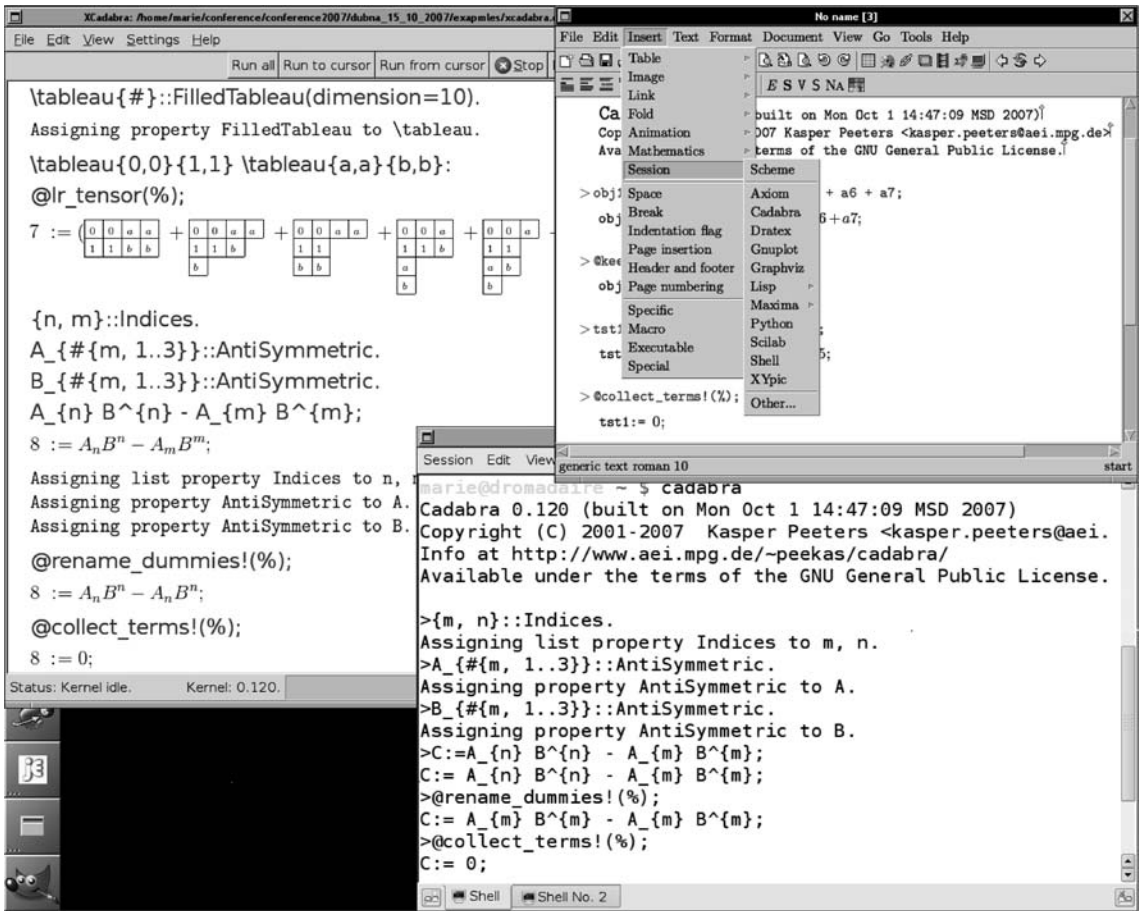
The program is completely independent of commercial software and relies only on a few other open source libraries and programs. Versions for Linux as well as for Mac OS X machines can be downloaded from the web site (http://www.aei.mpg.de/@~peekas/cadabra/index.html).

Cadabra can operate with Tensors, Spinors, Lie groups and as all CAS in symbolic view. It means that Cadabra has specials commands, specials data types from idem areas.

3. PECULIARITIES OF CADABRA

Let us observe several peculiarities of Cadabra [1, 2].

—*Usage of $\mathrm{T_EX}$ notation for both input and output, which eliminates the errors in transcribing problems from paper to computer and back.* The first and most easily visible feature is that every expression input has the form of a subset of $\mathrm{T_EX}$. Tensor indices, Dirac conjugation, derivative operators, commutators, the fer-



```
\tableau{#}::FilledTableau(dimension=10).
Assigning property FilledTableau to \tableau.
\tableau{0,0}{1,1} \tableau{a,a}{b,b}:
@lr_tensor(%);
7 := (
{n, m}::Indices.
A_{#{m, 1..3}}::AntiSymmetric.
B_{#{m, 1..3}}::AntiSymmetric.
A_{n} B^{n} - A_{m} B^{m};
8 := A_nB^n − A_mB^m;
Assigning list property Indices to n,
Assigning property AntiSymmetric to A.
Assigning property AntiSymmetric to B.
@rename_dummies!(%);
8 := A_nB^n − A_nB^n;
@collect_terms!(%);
8 := 0;
```

```
Ca                                   built on Mon Oct 1 14:47:09 MSD 2007)
Cop                                  007 Kasper Peeters <kasper.peeters@aei.mpg.de>
Ava                                  terms of the GNU General Public License.

>obj                                 + a6 + a7;
  obj                                6+a7;

>@kee
  obj

>tst1                                ;
  tst                                5;

>@collect_terms!(%);
  tst1:= 0;
```

```
marie@dromadaire ~ $ cadabra
Cadabra 0.120 (built on Mon Oct 1 14:47:09 MSD 2007)
Copyright (C) 2001-2007  Kasper Peeters <kasper.peeters@aei.
Info at http://www.aei.mpg.de/~peekas/cadabra/
Available under the terms of the GNU General Public License.

>{m, n}::Indices.
Assigning list property Indices to m, n.
>A_{#{m, 1..3}}::AntiSymmetric.
Assigning property AntiSymmetric to A.
>B_{#{m, 1..3}}::AntiSymmetric.
Assigning property AntiSymmetric to B.
>C:=A_{n} B^{n} - A_{m} B^{m};
C:= A_{n} B^{n} - A_{m} B^{m};
>@rename_dummies!(%);
C:= A_{m} B^{m} - A_{m} B^{m};
>@collect_terms!(%);
C:= 0;
```

Fig. 1. Cadabra in command line interface, xcadabra and Cadabra In T$_\mathrm{E}$X macs.

mion products and so on are all written just as in the T$_\mathrm{E}$X format. With a little bit of discipline, user can cut-and-paste expressions straight from a paper into a Cadabra notebook. The output, similarly, is typeset as T$_\mathrm{E}$X sequence, and Cadabra notebook files are in fact at the same time also valid T$_\mathrm{E}$X files (the program comes with a graphical notebook interface, but can also be used from the command line). Being able to input expressions like this would eliminate a large number of errors in transcribing physics problems to CAS.

—*An optional unlimited undo system. Interactive calculations can be undone to arbitrary level without requiring a full re-evaluation of the entire calculation.* We can infinitely many times return on each line of program, change something and restart it. But only command line interface in terminal dones not give this possibility.

—*A simple and documented way to add new algorithms in the form of C+ + modules, which directly operate on the internal expression tree.* The language that Cadadra was built on is C+ +, we could add some algorithms and functions in C+ +.

—*A command line interface as well as a graphical one, and a T$_E$X macs frontend.* Cadabra has 3 interfaces: command line interface, Xcadabra and T$_\mathrm{E}$X macs front-end. This picture (see Fig. 3) demonstrate it.

—*Built-in understanding of "dummy indices" and dummy symbols, including their automatic relabelling when necessary. Powerful algorithms for canonicalisation of objects with index symmetries, both mono-term and multi-term.* Cadabra operates indices: renames them, changes them.

—*A new way to deal with products of non-commuting objects, enabling a notation which is identical to standard physicist's notation (i.e. no need for special non-commuting product operators).* It's useful for non-commuting objects.

—*A flexible way to associate meaning ("type information") to tensors by attaching them to "properties."* In Cadabra there are many build-in types of tensor (tensor Weyl, tensor Rieman and so on).

4. EXAMPLES IN CADABRA

Let us consider 2 examples that illustrate Cadabra in action [1, 2].

Comparison of Cadabra and Maple

	Cadabra	Maple
1. Creator	Kasper Peeters	MapleSoft
2. First public release	2007	1985
3. Latest stable version	0.128 (November 28, 2007)	11.02 (November 26, 2007)
4. Open source	yes, with source code	no
5. License	GPL	proprietary
6. Operating system support	Mac OS X, Linux	Windows, Mac OS X, Linux
7. Language	C + +	C
8. Online help	no	yes
9. Graphics	no	yes
10. Modules support	yes	yes
11. Input	TEX notation and commands of system	commands of system
12. Output	TEX subset	TEX, HTML
13. Application	Quantum field theory, gravity, supergravity	Universal

4.1. Weyl Tensor

The first example is devoted to operations with Weyl tensor. Using the fact that Weyl tensor is built-in type we try to prove following identity.

$$\begin{aligned} W_{pqrs}W_{ptru}W_{tvqw}W_{uvsw} - W_{pqrs}W_{pqtu}W_{rutw}W_{svuw} \\ = W_{wnab}W_{npbc}W_{mscd}W_{spda} \\ -\frac{1}{4}W_{mnab}W_{psba}W_{mped}W_{nsdc}. \end{aligned} \qquad (1)$$

```
1  {m, n, p, q, r, s, t, u, v, w, a, b, c, d, e, f} ::Indices(vector).
2  W_{m n p q} ::WeylTensor.
3  W_{p q r s} W_{p t r u} W_{t v q w} W_{u v s w}
4   – W_{p q r s} W_{p q t u} W_{r v t w} W_{s v u w}
5   – W_{m n a b} W_{n p b c} W_{m s c d} W_{s p d a}
6   + (1/4) W_{m n a b} W_{p s b a} W_{m p c d} W_{n s d c};
```

We set indices of type vector in the first line. The second line vs determines type of Weyl tensor. We transpose all terms of our identity (lines 3–6) at the right side of equality. We have got in output:

$$\begin{aligned} 1 := W_{pqrs}W_{ptru}W_{tvqw}W_{uvsw} \\ - W_{pqrs}W_{pqtu}W_{rulw}W_{svuw} \\ - W_{mnab}W_{npbc}W_{mscd}W_{spda} \\ + 1/4\, W_{mnab}W_{psba}W_{mpcd}W_{nsdc}; \end{aligned} \qquad (2)$$

After that we preform the following operations.

```
1 @young_project_tensor!(%){ModuloMonoterm}:
2 @distribute!(%):
3 @canonicalise!(%):
4 @rename_dummies!(%):
5 @collect_terms!(%);
```

At the end we have got the result, where 2: means command number of system output:

$$2 := 0;$$

Let's discuss the sequnce of commands used. In the example we deal with rotation group SO(d). Young tables helps us working with group of symmetry (line 1). Command **distribute!** in line *2* removes brackets using dictribution rules. Line 3 we convert vs the expression to canonical form with the command **canonicalise!**. After that we rename indices (line 4), collect similar terms (line 5) and as a result we have got 0. Identity is approved.

4.2. Product of Gamma Matrices

Gamma Matrices are often used in operations with spinors. Cadabra has built-in algorithms, for example, algorithms for manipulation of gamma matrices in symbolic dimension. In the following example we expand the product $\gamma_{sr}\gamma_{rl}\gamma_{km}\gamma_{ms}$, as a result we aught to get the combination of γ_{kl} and δ_{kl} components.

$$\gamma_{st}\gamma_{rl}\gamma_{km}\gamma_{ms} = ?$$

```
1 ::PostDefaultRules(@@prodsort!(%), @@eliminate_kr!(%),
2                     @@canonicalise!(%), @@collect_terms!(%)).
3 {s,r,l,k,m,n} ::Indices(vector).
4 {s,r,l,k,m,n} ::Integer(0 .. d–1).
5 \gamma_{#} ::GammaMatrix(metric=\delta).
6 \delta_{m n} ::KroneckerDelta.
7 \gamma_{s r} \gamma_{r l} \gamma_{k m} \gamma_{m s};
8 join!(%){expand};
9 join!(%){expand};
```

In the first line there are some default simplification rules of the system which system will do after every stage. Let's write line 3–6 to declare the built-in types of objects: indices, their range, gamma matrix and Kronecker delta. After that we write initial product (line 7). Two times using the command **@join!** we expand the product of two adjacent gamma matrices.

$$1 := (-1)\gamma_{mr}\gamma_{lm}\gamma_{ks}\gamma_{rs};$$

$$2 := (-1)(2\gamma_{lr} - d\gamma_{lr} + \delta_{lr}d - \delta_{lr})\gamma_{ks}\gamma_{rs};$$

$$3 := (-1)(2\gamma_{lr} - \gamma_{lr}d + \delta_{lr}d - \delta_{lr}) \times (2\gamma_{kr} - d\gamma_{kr} + \delta_{kr} - \delta_{kr}d);$$

Further, as in first example let's use commands of reduction:

```
4 @distribute!(%);
5 @join!(%){expand};
6 @distribute!(%);
7 @factorise!(%){d};
8 @collect_factors!(%);
```

As a result after each command we have this expressions where the last is the desired one.

$$4 := -4\gamma_{lr}\gamma_{kr} + 4\gamma_{lr}\gamma_{kr}d + 4\gamma_{kl} - 6\gamma_{kl}d - \gamma_{lr}\gamma_{kr}dd + 2\gamma_{kl}dd - 2\delta_{kl}d + \delta_{kl}dd + \delta_{kl};$$

$$5 := 12\gamma_{kl} - 4d\gamma_{kl} - 3\delta_{kl} + 2\delta_{kl}d + 4(-2\gamma_{kl} + d\gamma_{kl} + \delta_{kl} - \delta_{kl}d)d - 6\gamma_{kl}d - (-2\gamma_{kl} + d\gamma_{kl} + \delta_{kl} - \delta_{kl}d)dd + 2\gamma_{kl}dd + \delta_{kl}dd;$$

$$6 := 12\gamma_{kl} - 18\gamma_{kl}d - 3\delta_{kl} + 6\delta_{kl}d + 8\gamma_{kl}dd - 4\delta_{kl}dd - \gamma_{kl}ddd + \delta_{kl}ddd;$$

$$7 := \gamma_{kl}(12 - 18d + (8(dd) - (ddd))) + \delta_{kl}(-3 + 6d + (-4(dd) + (ddd)));$$

$$8 := \gamma_{kl}(12 - 18d + 8d^2 - d^3) + \delta_{kl}(-3 + 6d - 4d^2 + d^3);$$

5. COMPARISON OF CADABRA AND MAPLE

At the end let's compare two CAS: Cadabra and Maple. Results of comparison are shown in the table where documentation and particular experience were used. Besides, information from source http://en.wikipedia.org/wiki/Comparison_of_computer_algebra_systems was considered. Maple is a well-known and widespread system working from 1985. Cadabra was created in 2006 and was developed by Kasper Peeters. It was made for some scientific problems only, but was made profoundly.

Latest stable versions of Maple and Cadabra were released on April 26th 2009 and July 18th 2009, the systems evolve and are upgrading. The advantage of Cadabra is that it's distributed under GPL and its source code is open. This fact is very important in Russia nowadays. Particularly, in Peoples' Friendship University of Russia free operation systems become more popular. From this point of view Cadabra works with Linux, Mac OS X and has been ported on Windows. Maple support all operation systems, but even the student's version of Maple costs about \$100, and the full version costs 20 times more.

Cadabra as Maple has possibility to add modules, but hasn't graph support. Another one Cadabra disadvantage is absence of interactive help.

One of Cadabra advantages is that input and output are in $\TeX$ notation. That can be useful for writing scientific articles and books. To insert a formula or result of calculations you simply need to cut it from Cadabra window and paste into the text of article.

The next point of comparison is an area of applications. Maple is a universal system which contains many algorithms of calculation methods. Cadabra's area of applications is limited to problems in field theory.

We have seen some examples of Cadabra in Section 4. It turned out that it's impossible to solve this examples in Maple in symbolic form because Maple has the followind problems:

—It's impossible to operate with Weyl tensors in symbolic form.

—It's possible to operate with gamma matrices of one indice only.

—The dimension of space must be defined numericaly and cann't in be symbolic from for example *d.*

—Maple operates only with matrices and their components, but not with tensors in general form. Whereas Cadabra can work with abstract component tensors as well was with individual components.

CONCLUSIONS

Having compared the two systems we conside that they have different paradigms. Maple can be successfully used in *general engineering problems.* It was built to solve equation systems and in consequence ofthis it permits working with matrices. To manipulate with tensors we need to represent all its components in matrices too. To represent a symmetric or antisymmetric tensor we need to symmetrize or antisymmetrize the corresponding matrix. Cadabra was designed for *scientists* dealing with *physical problems.* It works with objects of field theory in form of abstract components and numerical.

At the end, we recommend Cadabra CAS as a useful tool for scientists in their physical researches.

REFERENCES

1. K. Peeters, "Introducing Cadabra: A Symbolic Computer Algebra System for Field Theory Problems," hep-th/0701238; Jaunary 2007, http://www.aei.mpg.de~peekas/cadabra/cadabra_hep.ps.

2. K. Peeters, "Cadabra: Reference Guide and Tutorial," Preprint AEI-2006-038 (2001–2007); http://www.aei.mpg.de~peekas/cadabra/cadabra.ps.