Prospects of FORM

J.A.M. Vermaseren

- Introduction
- Status and examples
- Perceived needs
- What is realistic
- Conclusions

1 Introduction

Computer algebra is a necessary tool in the field of perturbative field theory. As the accelerators get more and more luminosity there is a greater need for more precise calculations in order to have a comparable precision in the experiments and the theoretical predictions. If for instance a TESLA-like machine will be built in which accuracies in the order of 10^{-3} can be reached, the current calculations will nearly all have to be extended by one loop. This means that the amount of algebra to be done will increase by several orders of magnitude over what was used for the current calculations, and already those used nonnegligible amounts of computer resources. There exist several good examples:

- The recent calculation of QCD structure functions for deep inelastic scattering to three loops took several CPU years on top level workstations. Integral tables were constructed that took more than 3 Gbutes. Even big pieces of the eventual programs were derived with the help of computeralgebra. More future QCD calculations may be like this.

- The GRACE system for the automatic calculation of one loop processes in electroweak interactions uses diagram by diagram algebraic reductions that construct via the computer algebra numerical programs for the eventual integrations. Typically each reaction has very many diagrams and each takes its time. Extension to two loops would be a real challenge and it is not clear whether computer resources for it would be available.

- One popular method to deal with Feynman diagrams with loops is the algebraic reduction to masterintegrals. This involves either a complicated reduction scheme as was used in the first example or finding a solution to a very large sparse set of linear equations. As field theory tends to create large (numerical) cancellations between diagrams the solution of the equations is done algrabraically after which each diagram can be expressed as a linear combination of master integrals. The coefficients may be rational functions of one or more parameters.

It should be clear from this that the future will bring us an ever greater need for massive amounts of computer algebra resources.

The above situation is not unique to perturbative field theory (PFT). Also other fields of science have discovered that much original research can be done with the use of computeralgebra. Actually, the most popular 'integrated' packages are not designed with PFT in mind. They serve more like mathematical workbenches and cater to the somewhat smaller needs. Yet, with the advent of more powerful computers also these systems have enormous processing capacities. One rather popular way to use such systems is to expand a series to many more terms than one would be able to do by hand and then guess or fit the pattern. After finding the 'correct' formula it is often much easier to prove it than to derive it from first principles.

In this talk we will concentrate on one specific system and see what is needed to make it even more useful and have it keep up with modern trends in symbolic calculations.

2 Status and examples

Before we can have a look at a future FORM we should look at FORM now. What are the things that FORM can do?

- It expands formulae.
- It can handle formulae as large as the available disk space.
- It can deal with vectors, indices, tensors, functions.
- It makes substitutions via sophisticated pattern matching.
- It has many special built in functions.
- It can control the order of the output.
- It has a preprocessor.
- It has special variables that communicate between the algebra and the preprocessor, making the programming very flexible.
- It has an extensive facility for tables.
- It can be parallelized.
- and much more....

The net result is a program that can be used for most operations needed in scientific calculations that involve expansions. Many problems can be written as such, although sometimes it requires a lot of ingenuity. The result on the other hand is usually a very fast program that can handle very big problems.

The past few years development has been dominated by the needs for the calculation of deep inelastic structure functions in QCD at the three loop level. This was a rather horrendous calculation that was only barely possible with the current computer resources and the capabilities of FORM. Special facilities had to be provided, like the management of very large tables. As the calculations in perturbative field theory for the LHC will move to the NNLO level these facilities will be welcomed by other groups as well.

As the most important of these three loop calculations have been finished now, it becomes time to start looking around again.

```
    Symbols a,b,c;
    Local F = (a+b+c)^10;
    Format 55;
    Print;
    .end

Time =       0.00 sec    Generated terms =         66
                F        Terms in output =         66
                         Bytes used      =       1160

   F =
      c^10 + 10*b*c^9 + 45*b^2*c^8 + 120*b^3*c^7 + 210
      *b^4*c^6 + 252*b^5*c^5 + 210*b^6*c^4 + 120*b^7*
      c^3 + 45*b^8*c^2 + 10*b^9*c + b^10 + 10*a*c^9 +
      90*a*b*c^8 + 360*a*b^2*c^7 + 840*a*b^3*c^6 +
      1260*a*b^4*c^5 + 1260*a*b^5*c^4 + 840*a*b^6*c^3
       + 360*a*b^7*c^2 + 90*a*b^8*c + 10*a*b^9 + 45*
      a^2*c^8 + 360*a^2*b*c^7 + 1260*a^2*b^2*c^6 +
      2520*a^2*b^3*c^5 + 3150*a^2*b^4*c^4 + 2520*a^2*
      b^5*c^3 + 1260*a^2*b^6*c^2 + 360*a^2*b^7*c + 45*
      a^2*b^8 + 120*a^3*c^7 + 840*a^3*b*c^6 + 2520*a^3
      *b^2*c^5 + 4200*a^3*b^3*c^4 + 4200*a^3*b^4*c^3
       + 2520*a^3*b^5*c^2 + 840*a^3*b^6*c + 120*a^3*
      b^7 + 210*a^4*c^6 + 1260*a^4*b*c^5 + 3150*a^4*
      b^2*c^4 + 4200*a^4*b^3*c^3 + 3150*a^4*b^4*c^2 +
      1260*a^4*b^5*c + 210*a^4*b^6 + 252*a^5*c^5 +
      1260*a^5*b*c^4 + 2520*a^5*b^2*c^3 + 2520*a^5*b^3
      *c^2 + 1260*a^5*b^4*c + 252*a^5*b^5 + 210*a^6*
      c^4 + 840*a^6*b*c^3 + 1260*a^6*b^2*c^2 + 840*a^6
      *b^3*c + 210*a^6*b^4 + 120*a^7*c^3 + 360*a^7*b*
      c^2 + 360*a^7*b^2*c + 120*a^7*b^3 + 45*a^8*c^2
       + 90*a^8*b*c + 45*a^8*b^2 + 10*a^9*c + 10*a^9*b
       + a^10;
```

```
    Symbols a,b,c,d,e,f,g,h;
    L F = (a+b+c+d+e+f+g+h)^30;
    .end

Time =       0.27 sec    Generated terms =      50000
               F      1 Terms left      =      50000
                        Bytes used      =    1603470
                      .
                      .
                      .
                      .
Time =     131.67 sec
               F        Terms active    =   10295472
                        Bytes used      =  362052754

Time =     136.04 sec    Generated terms =   10295472
               F        Terms in output =   10295472
                        Bytes used      =  361695538
```

```
    #: SmallSize 20M
    #: LargeSize 200M
    #: TermsInSmall 1M
    Symbols a1,...,a10;
    Local F = (a1+...+a10)^30;
    .end

Time =       3.96 sec    Generated terms =     432993
               F      1 Terms left      =     432993
                        Bytes used      =   15338730
                      .
                      .
                      .
                      .
                      .
Time =    3595.46 sec
               F        Terms active    =  211915132
                        Bytes used      = 8386638124

Time =    3736.78 sec    Generated terms =  211915132
               F        Terms in output =  211915132
                        Bytes used      = 8380179758
```

```
    CFunction f;
    Symbol x;
    Local F = f(x)+f(x^2)+f(x,x+1)+f;
    Print;
    .sort

Time =       0.00 sec    Generated terms =          4
               F         Terms in output =          4
                         Bytes used      =        114

   F =
      f + f(x^2) + f(x) + f(x,1 + x);

    Functions A,B;
    Drop F;
    Local G = (A+B)^3;
    Print;
    .end

Time =       0.00 sec    Generated terms =          8
               G         Terms in output =          8
                         Bytes used      =        162

   G =
      A*A*A + A*A*B + A*B*A + A*B*B + B*A*A + B*A*B
       + B*B*A + B*B*B;
```

```
    Symbols x,n,a,b,c,d,e;
    Set abc:a,b,c;
    CFunction f;
    Local F = f(a)+f(b)+f(c)+f(d)+f(e);
    id f(x?abc[n]) = f(x,n);
    Print;
    .end

Time =       0.00 sec    Generated terms =          5
                F        Terms in output =          5
                         Bytes used      =         86

   F =
      f(a,1) + f(b,2) + f(c,3) + f(d) + f(e);
```

```
    Symbols x,y,z;
    Indices i1,...,i6;
    CFunction I;
    Local G = I(i1,i2,x)*I(i2,i3,y)*I(i3,i4,z)
             *I(i4,i5,x)*I(i5,i6,z)*I(i6,i1,y);
    repeat;
      id  I(i1?,i2?,?a)*I(i2?,i3?,?b) = I(i1,i3,?a,?b);
    endrepeat;
    Print;
    .end

Time =       0.00 sec    Generated terms =          1
               G         Terms in output =          1
                         Bytes used      =         48

   G =
      I(i1,i1,x,y,z,x,z,y);
```

```
    CF den;
    S  x,x1,x2;
    L  F = den(x+1)*den(x+2)^2*den(x+3)^3*den(x+4)^4;
    SplitArg,den;
    Print;
    .sort
Time =       0.00 sec    Generated terms =          1
               F         Terms in output =          1
                         Bytes used      =        150
   F =
      den(1,x)*den(2,x)^2*den(3,x)^3*den(4,x)^4;

    repeat;
      id  den(x1?!{x2?},x)*den(x2?!{x1?},x) =
     (den(x1,x)-den(x2,x))*den(x2-x1);
    endrepeat;
    id den(x?number_) = 1/x;
    Print +s;
    .end
Time =       0.00 sec    Generated terms =        181
               F         Terms in output =         10
                         Bytes used      =        216

   F =
       + 1/648*den(1,x)
       + 1/4*den(2,x)
       - 1/16*den(2,x)^2
       - 17/8*den(3,x)
       + 3/4*den(3,x)^2
       - 1/2*den(3,x)^3
       + 607/324*den(4,x)
       + 403/432*den(4,x)^2
       + 13/36*den(4,x)^3
       + 1/12*den(4,x)^4
      ;
```

```
#procedure normden2(x,den)
SplitArg,((‘x’)),‘den’;
id ‘den’(‘x’) = ‘den’(0,‘x’);
#do inormden = 1,1
 id,once,ifmatch->1,‘den’(x1?!{x2?},‘x’)*
                  ‘den’(x2?!{x1?},‘x’) =
    (‘den’(x1,‘x’)-‘den’(x2,‘x’))*
                  ‘den’(x2-x1);
 goto 2;
 Label 1;
          redefine inormden "0";
 Label 2;
 id ‘den’(x1?number_) = 1/x1;
 .sort:normden;
#enddo
id  ‘den’(x1?,x2?) = ‘den’(x1+x2);
#endprocedure
```

```
    #define MAX "10"
    CF yden;
    S  y,x1,x2,n1,n2,i;
    L  F = <yden(y+(1))^1>*...*<yden(y+(`MAX'))^`MAX'>;
    #call normden2(y,yden)
                        .
                        .
Time =       0.07 sec    Generated terms =         55
                F        Terms in output =         55
                 normden Bytes used      =       2356
    Format 55;
    Print +s;
    .end
Time =       0.07 sec    Generated terms =         55
                F        Terms in output =         55
                         Bytes used      =       3420
   F =
       + 1/7830203310981140781182893575634944000000*
      yden(1 + y)
       + 1741/126764165966642293354751066112000000000*
      yden(2 + y)
       - 1/90545832833315923824822190080000000*yden(2
       + y)^2
       - 2354081/24990587170998417453219840000000000*
      yden(3 + y)
       + 191/1322253289470815738265600000000*yden(3
       + y)^2
       - 1/85001997180266726031360000000*yden(3 + y)^
      3
       - 578201/2339941401640304640000000000*yden(4
       + y)
       + 3229/584985350410076160000000000*yden(4 + y)^
      2
       - 43/4874877920083968000000000*yden(4 + y)^3
       + 1/1299967445355724800000000*yden(4 + y)^4
       + 710039261/53496602689536000000000000*yden(5
```

```
 + y)
 - 325111/8358844170240000000000000*yden(5 + y)^2
 + 41269/4458050224128000000000000*yden(5 + y)^3
 - 1/6191736422400000000*yden(5 + y)^4
 + 1/61917364224000000000*yden(5 + y)^5
 + 82050111529/11832592569282330624000000*yden(6
 + y)
 - 1995602407/7888395046188220416000000*yden(6 +
y)^2
 + 991031/1232561725966909440000*yden(6 + y)^3
 - 23389/1095610423081697280000*yden(6 + y)^4
 + 13/3043362286338048000*yden(6 + y)^5
 - 1/182601737180282880*yden(6 + y)^6
 - 2619577285421/19499511680335872000000000*yden(
7 + y)
 + 342118488383/5849853504100761600000000*yden(7
 + y)^2
 - 468816559/194995116803358720000000*yden(7 + y)^
3
 + 6838399/8124796533473280000000*yden(7 + y)^4
 - 9289/338533188894720000*yden(7 + y)^5
 + 281/45137758519296000*yden(7 + y)^6
 - 1/752295975321600*yden(7 + y)^7
 - 1065694909851464692912335890747249592893440\
000000000*yden(8 + y)
 + 168020328664835851/881135053374970920960000000\
000*yden(8 + y)^2
 - 38401012214437/2097940603273740288000000000*
yden(8 + y)^3
 + 41448696407/8880171865708953600000000*yden(8 +
y)^4
 - 12749591/4757234928058368000000*yden(8 + y)^5
 + 929083/1132674982871040000000*yden(8 + y)^6
 - 1/499415777280000*yden(8 + y)^7
 + 1/128421199872000*yden(8 + y)^8
 + 184010862703357932414039121/83560363308089402\
```

```
3578681344000000000*yden(9 + y)
 + 10341739293083887023479 3/9947662298582071709 2\
700160000000000*yden(9 + y)^2
 + 116879994120931549433/25376689537199162523648\
0000000000*yden(9 + y)^3
 + 131743320840433661/70490804269997673676800000\
0000*yden(9 + y)^4
 + 3875529938683/55945082753966407680000000*
yden(9 + y)^5
 + 13350056123/59941160093535436800000 0*yden(9
 + y)^6
 + 98423/1585744976019456000 0*yden(9 + y)^7
 + 67/52438656614400000*yden(9 + y)^8
 + 1/5056584744960000*yden(9 + y)^9
 - 1325123693480321087275613652699623401/
752012725986628760624805099003980021760000000000\
000*yden(10 + y)
 - 3813320064163568403976474755184 49/59683549681\
4784730654607221431730176000000000000*yden(10 +
y)^2
 - 1256253590075611030603504204 1/592098707157524\
5343795706561822720000000000 0*yden(10 + y)^3
 - 8531997282884746751079 77/13426274538719377196\
8156611379200000000 0*yden(10 + y)^4
 - 2240921699094855930 41/13319716804285096425412\
362240000000000*yden(10 + y)^5
 - 56678105555600429/14799685338094551583791513 6\
00000000*yden(10 + y)^6
 - 528480938221/7341113758975472015769600000 00*
yden(10 + y)^7
 - 244274843/2330512304436657782784000000*yden(
10 + y)^8
 - 4861/4624032350072733696000 00*yden(10 + y)^9
 - 1/1834933472251084800000*yden(10 + y)^10
;
```

```
    #define MAX "4"
    Symbols x1,...,x{2*‘MAX’+1};
    CF f;
    Local F = f(x1,...,x{2*‘MAX’+1});
    Print;
    .end

Time =       0.00 sec    Generated terms =          1
                F        Terms in output =          1
                         Bytes used      =         52

   F =
      f(x1,x2,x3,x4,x5,x6,x7,x8,x9);
```

```
    Symbols x,y,n;
    L F = (x+y)^3+(x-y+2)^4;
    id x^n? = x^(n+1)/(n+1);
    .sort

Time =       0.00 sec    Generated terms =         19
                F        Terms in output =         15
                         Bytes used      =        234
    Format doubleFortran;
    #write <fun.f> "      REAL*8 FUNCTION fun(x,y)"
    #write <fun.f> "      REAL*8 x,y"
    #write <fun.f> "*\n*      Routine created by FORM, `DATE_'\n*"
    #write <fun.f> "      fun = %E",F
    #write <fun.f> "      RETURN"
    #write <fun.f> "      END"
    Print +f;
    .end

Time =       0.00 sec    Generated terms =         15
                F        Terms in output =         15
                         Bytes used      =        234

      F =
     & 16.D0*x - 32.D0*x*y + 24.D0*x*y**2 - 7.D0*x*y**3 + x*y**4 + 16.D0
     & *x**2 - 24.D0*x**2*y + 27.D0/2.D0*x**2*y**2 - 2.D0*x**2*y**3 + 8.
     & D0*x**3 - 7.D0*x**3*y + 2.D0*x**3*y**2 + 9.D0/4.D0*x**4 - x**4*y
     &  + 1.D0/5.D0*x**5
```

```
      REAL*8 FUNCTION fun(x,y)
      REAL*8 x,y
*
*       Routine created by FORM, Fri May 20 11:07:02 2005
*
      fun = 16.D0*x - 32.D0*x*y + 24.D0*x*y**2 - 7.D0*x*y**3 + x*y**4
     &  + 16.D0*x**2 - 24.D0*x**2*y + 27.D0/2.D0*x**2*y**2 - 2.D0*x**2*
     & y**3 + 8.D0*x**3 - 7.D0*x**3*y + 2.D0*x**3*y**2 + 9.D0/4.D0*x**4
     &  - x**4*y + 1.D0/5.D0*x**5
      RETURN
      END
```

```
    Symbols x,y,z;
    L F = (x+y+z)^5-(x+2*y)^5;
    .sort

Time =       0.00 sec    Generated terms =         27
                F        Terms in output =         20
                         Bytes used      =        364
    #$xc = 0;
    if ( count(x,1) > $xc ) $xc = count_(x,1);
    .sort

Time =       0.00 sec    Generated terms =         20
                F        Terms in output =         20
                         Bytes used      =        364
    #message The maximum power of x is `$xc'
~~~The maximum power of x is 4
    Print +f;
    Format 55;
    Bracket x;
    .end

Time =       0.00 sec    Generated terms =         20
                F        Terms in output =         20
                         Bytes used      =        386

   F =
       + x * ( 5*z^4 + 20*y*z^3 + 30*y^2*z^2 + 20*y^3*
         z - 75*y^4 )
       + x^2 * ( 10*z^3 + 30*y*z^2 + 30*y^2*z - 70*y^3
          )
       + x^3 * ( 10*z^2 + 20*y*z - 30*y^2 )
       + x^4 * ( 5*z - 5*y )
       + z^5 + 5*y*z^4 + 10*y^2*z^3 + 10*y^3*z^2 + 5*
         y^4*z - 31*y^5;
```

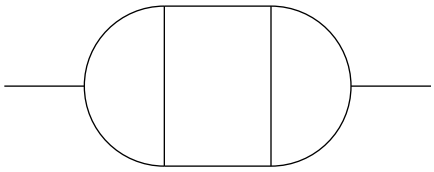
```
    Symbols x,y;
    Table ch(0:6,x?);
    Fill ch(0) = 1;
    Fill ch(1) = x;
    Fill ch(2) = x*ch(1,x)+(x+1)*ch(0,x);
    Fill ch(3) = x*ch(2,x)+(x-1)*ch(1,x);
    Fill ch(4) = x*ch(3,x)+(x+1)*ch(2,x);
    Fill ch(5) = x*ch(4,x)+(x-1)*ch(3,x);
    Fill ch(6) = x*ch(5,x)+(x+1)*ch(4,x);
    Local F5 = ch(5,y);
    Local F6 = ch(6,y);
    Print +f;
    .end

Time =       0.00 sec    Generated terms =         21
               F5        Terms in output =          4
                         Bytes used      =         54

Time =       0.00 sec    Generated terms =         43
               F6        Terms in output =          7
                         Bytes used      =         86

   F5 =
      y + 3*y^3 + 4*y^4 + y^5;

   F6 =
      1 + 3*y + 5*y^2 + 5*y^3 + 7*y^4 + 5*y^5 + y^6;
```



```
    #define TOPO "la"
    #define SCHEME "0"
    #include- mincer.h
    off statistics;
    .global
    Local F = Q.Q^10/p1.p1^2/p2.p2^2/p3.p3^2/p4.p4^2
              /p5.p5^2/p6.p6^2/p7.p7^2/p8.p8^2;
    Multiply ep^3;
    #call integral(‘TOPO’)
~~~Answer in MS-bar
    .sort
    On Statistics;
    Print +s;
    .end

Time =       0.47 sec    Generated terms =          5
               F         Terms in output =          5
                         Bytes used      =         80

   F =
       - 389662969/13500
       - 4964/3*ep^-3
       - 80739/5*ep^-2
       - 56411291/1350*ep^-1
       + 325708/3*z3
      ;
```

3 Perceived Needs

The question here is not so much what FORM can do, but what it cannot do and would be needed to make it even more powerful.

The first of these needs is something called factorization and/or simplification. FORM always expands formulae, taking away its brackets. This does result in an unique output that can be easily manipulated, but sometimes such an output is unnecessaily lengthy. The first improvement would be factorization. This is to split the output into two or more multiplicative factors like in $(a+b)(c+d)$.

There is a whole theory about how to set up factorization, but if there are many variables and the formula is very large, this can become rather slow. The basic idea is to first make sure that all coefficients are integer. In case of fractions one can multiply the whole formula with the least common multiple of the denominators. Next one decides to work over the numbers modulus a prime number. This is done several times for different prime numbers so that the product of the prime numbers used is larger than the largest integer in the formula. Out of the various factorizations one can then reconstruct the factorization over all integers, using the chinese remainder theorem.

The factorization modulus prime numbers is done by giving all variables except for one a value. This gives a polynomial over one variable which can be factorized, using some high level mathematics. Next one gives one of the variables that was fixed a different value and one factorizes again. Doing this for several values, one can reconstruct the complete factorization in two variables. Then one changes the value of the third variable and does the factorization in the remaining two variables again (by giving the second variable different values again) etc. As one can see, this solution goes exponential in the number of variables in the formula. Fortunately in many cases there

are heuristic tricks that can help, like in the case that a formula is linear in a given variable.

The above may look complicated but is nothing compared to general simplification which writes a formula generically in the form $a\ b + c\ d$. Such an operation is usually not unique and there is no proper mathematical solution for it. Usually this is done by just looking whether one may spot subexpressions that occur more than once. Success is not guaranteed.

It should be clear however that factorization of a formula with millions of terms would bring enormous savings w.r.t. the length of the output. It would also make it easier to interprete the mathematical contents of the output. But at the same time it should be equally clear that it will be a large amount of work to built in a factorization/simplification system that can handle millions of terms.

One of the applications of factorization-like algorithms is solving large sets of equations in many variables in which the coefficients are polynomials in a number of parameters. The usual techniques for this work with cross multiplications and it is rather important to recognize common factors in order to avoid the coefficients from becoming more complicated than necessary. Actually for this one needs mostly good algorithms for finding Greatest Common Divisors. The best algorithms for this work however in a way similar to the algorithms for factorization: by giving the parameters values and solving the real problem for the GCD in single variables only. Currently FORM can only efficiently handle the solution of massive systems of linear equations in which the coefficients are numerical.

In the field of solving systems of equations there is also a need for solving systems of nonlinear equations. The way these can be solved by standard method is by means of Gröbner bases. In all but the simplest cases this method of solving has the tendency to give extremely large intermediate results. As these formulae are of a polynomial nature, FORM should be very good at this.

The implementation of factorization/simplification algorithms and Gröbner bases leads to a large amount of work, even if existing packages are being used as a means to get started. Existing packages will not use the FORM methods of storing polynomials. Yet this seems to be the best way to proceed.

Assume the set of equations

$$\begin{aligned} 1 &= x\ y + y\ z + z\ x \\ 4 &= x^2 + y^2 + z^2 \\ 3 &= x^2 + x\ y - x\ z - y^2 \end{aligned}$$

By making subtractions and additions one can show that the solution space of this set of equations is equal to the solution space of the following set of equations

$$\begin{aligned} 1 &= y\ z + x\ z + x\ y \\ 0 &= -z^3 + 2\ y - 3\ y^2\ z - 2\ y^3 \\ 0 &= 64\ z^3 - 912\ z^5 + 3105\ z^7 - 918\ z^9 + 81\ z^{11} \end{aligned}$$

This modified set of equations, called a Gröbner basis, can be used to read off the solutions with a minimal amount of effort.

Another request of users concerns something of the type subroutines with their own local variables. The current situation is that FORM has 'procedures' that are actually giant macro's and that are used like subroutines, but the 'local' variables all go into the global namespace and hence there are in reality no local variables. What is needed for real subroutines? The first issue is whether the routine should be translated once for many uses. If this is the case, the local variables need to be relocated and hence a whole convention for relocation needs to be developed. And this system should be capable of distinguishing between local and global variables. This can only be done if the language insists on the declaration of all variables that are used as either local or global. This leaves lots of room for problems and errors. Another complication occurs when, on exit, local variables survive in the output formulae. Such an occurrence is a fatal error, comparable to an undefined variable in a computational language.

The complexity of a proper implementation of real subroutines may be the cause of a shortcut in this direction: a preprocessor command to define a local namespace in part of the program. This offers the major benefit of a subroutine at a fraction of the difficulties. This subject is however still in the planning stage.

Another useful feature would be a proper LaTeX output mode. For this it is necessary to have a LaTeX representation for each variable that can occur in the output and a system for calculating the line length. This should not be excessively difficult.

A very useful feature would be the existence of proper communication channels with programs like Maple or Mathematica. Such channels would allow FORM to give certain tasks to one of those programs et vise versa. In principle the protocols exist (OpenMath and MathLink) and people have also experimented with automatically moving formulae between the systems in a way that is controled by a separate class of programs. The task is to do this in a structured way and to give FORM more tools to make this easier.

Another feature that is missed by many is more documentation. Currently there exist a tutorial (written by A. Heck) and a reference manual. The tutorial gives a good introduction to FORM but, due to its limited length, cannot go into the fine details. The reference manual is mainly concerned with giving complete and correct information, but it is much harder to be read. A third part of the manual should be a compendium of worked out examples that should show the actual use of FORM by experts, explained for beginners. Especially the path to optimizing solutions for given problems should be explained, helping many users in getting the best out of FORM.

To allow others to make contributions to the sources of FORM, its inner conventions and workings should be explained in a separate manual. This should be a rather big manual and it will take much work to write it.

With the slowdown in the development of faster and faster processors the parallelization of FORM becomes ever more important. The currently available program runs only on a limited number of machines and it will be very important to have versions for systems with two or four processors as well as for systems with a very large number of processors. This seems to be mainly a matter of available manpower as the principle by which to do it has already be studied. This parallelization gives FORM an extra advantage over other systems, as currently neither Maple nor Mathematica can be parallelized 'in completo'.

More libraries for the solution of more or less standard tasks would be very useful for users as well. There exist some libraries (like Mincer, summer and harmpol) but there is a need for several others (like solving systems of linear equations).

4 What is realistic

Of course it is one thing to make an extensive whish list of what one would like to build in. It is a completely different thing to see what can be obtained realistically in the forseeable future. Over the past years an attempt has been made to acquire funds for the implementation of all of the above. The amount of work was estimated at 20 manyears. Things looked good at the beginning and actually during most of the process. Just when the money was going to be awarded though the main sponsor suddenly declared that they were out of money. Since then only money has been obtained for the parallelization effort, for the manual of the internal conventions and procedures and for the creation of some specific applications.

The development of the manual of the internal functioning of FORM is very important. It should allow eventually FORM to become an open source program. The original plan was to rewrite FORM first in order to insure the coherency of its internals. At the current level of financing this is unrealistic. It is a pity as the current sources are the result of a 20 years effort and hence have seen several generations of attitudes about how to program properly. Also in the beginning FORM was on a much more modest scale and most of the later features had not yet been forseen.

The first effort now should go to prepare FORM to be an open source program. This involves, as mentioned, the preparation of much documentation, and possibly the reprogramming of some parts of extremely confusing code. Some parts of the old code of FORM have seen so many generations of ad hoc 'improvements' that they are very difficult to read or to debug.

The next on the list should be the version for computers with two processors. To have FORM perform with almost twice the speed of a single processor on such machines would be a real benefit as such machines are nowadays relatively cheap.

After this the most important feature would be the factorization. The best approach seems to be to take the best available library for this and to see how it can be modified to make it coherent with FORM. This might be of benefit also to the original author of that library as the memory management of FORM could leave this author with a more powerful library and working environment as before. One currently very good library comes together with a system for finding Gröbner bases. This would kill two birds with one stone.

It may not be easy to get the funds for the factorization effort. As mentioned before, there exists already a mathematical theory as to how to do this, so from the mathematical viewpoint this isn't an interesting problem any longer. Hence we cannot realistically expect to much support from that direction. At the same time it isn't a physics problem either. The only money one can hope for is from a source that is interested in applied science or interdisciplinary science. Maybe it will be possible to look for EU funds in a future round of applications.

5 Conclusions

FORM is alive and kicking. The most needed extensions need however a sizable amount of work. It is not clear how to obtain funds for this as the most natural sponsor claims a lack of money.