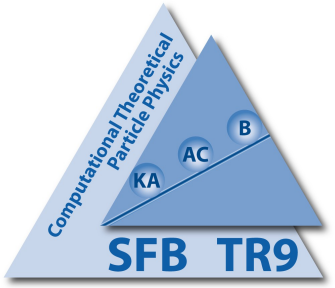
Recent developments on FORM

Takahiro Ueda
TTP KIT Karlsruhe, Germany

Jos Vermaseren
Nikhef, The Netherlands



ACAT 2013
16 May 2013, IHEP, Beijing

Outline

- Introduction of FORM
- Recent developments after FORM 4.0 (→ ver. 4.1)
  - Code optimization (including Monte Carlo Tree Search)

    J. Kuipers, J.A.M. Vermaseren, A. Plaat, H.J. van den Herik, arXiv:1207.7079 [cs.SC].

    J. Kuipers, J.A.M. Vermaseren, in preparation.
- Conclusion

What is FORM?

- FORM is a program for symbolic manipulation of mathematical expressions. (Jos Vermaseren et al.)
- Open source: http://www.nikhef.nl/~form
- Designed to handle huge expressions with an arbitrary number of terms. Not limited by the memory, but only by the disk space.
- Basic tool for performing large scale calculations in perturbative quantum field theory.
- Parallel versions:
  - TFORM : POSIX threads, for multi-core processors.
  - ParFORM : MPI, on computer clusters, etc.

FORM 4.0

- The current official version is FORM version 4.0.

  J. Kuipers, T. Ueda, J.A.M. Vermaseren and J. Vollinga,
  CPC 184 (2013) 1453, arXiv:1203.6543 [cs.SC]

- Polynomial factorization.

  FactArg, Factorize, #FactDollar, ...

- Play with rational polynomials.

  PolyRatFun, gcd_, div_, rem_

  - Release of version 4.0 on 29 March 2012.
  - Executables with some bug fixes on 10 April 2012.
  - Further developments and bug fixes in the CVS repository...

Code Optimization

- Needs to export analytical results of FORM to Fortran or C etc. code for further numerical computations.
- If a lengthy expression is to be evaluated millions of times (e.g., Monte Carlo integration), it is worth reducing the code size even if this needs a non-negligible time.
- In the next version of FORM, Format statement accepts options for the code optimization. Then Print statement prints the optimized code. (In practice, one can use #Optimize and #Write for writing the optimized code into files.)
- For user convenience, we provide O0, O1, O2, O3 options, a certain combinations of options and parameter settings.

```
Symbols x,y,z;
ExtraSymbols vector,v;
Local F = 6*y*z^2 + 3*y^3 - 3*x*z^2 + 6*x*y*z - 3*x^2*z
          + 6*x^2*y;
Format Fortran;
Format O1,stats=on;
Print;
.end
```

- The input is a polynomial of 3 variables x, y and z.
- w/ O1, # of ops.: 23 → 15

```
      v(1)=y*z
      v(2)= - z + 2*y
      v(2)=x*v(2)
      v(3)=z**2
      v(1)=v(2) - v(3) + 2*v(1)
      v(1)=x*v(1)
      v(2)=y**2
      v(2)=2*v(3) + v(2)
      v(2)=y*v(2)
      v(1)=v(2) + v(1)
      F=3*v(1)
*** STATS: original  1P 16M 5A : 23
*** STATS: optimized 0P 10M 5A : 15
```

P: powers
M: multiplications (incl. squaring)
A: additions

One power counted double because y^3 = y * y * y.

```
Symbols x,y,z;
ExtraSymbols vector,v;
Local F = 6*y*z^2 + 3*y^3 - 3*x*z^2 + 6*x*y*z - 3*x^2*z
          + 6*x^2*y;
Format Fortran;
Format O2,stats=on;
Print;
.end
```

- The input is a polynomial of 3 variables x, y and z.
- w/ O2, # of ops.: 23 → 14

```
      v(1)=z**2
      v(2)=2*y
      v(3)=z*v(2)
      v(2)= - z + v(2)
      v(2)=x*v(2)
      v(2)=v(2) - v(1) + v(3)
      v(2)=x*v(2)
      v(3)=y**2
      v(1)=2*v(1) + v(3)
      v(1)=y*v(1)
      v(1)=v(1) + v(2)
      F=3*v(1)
*** STATS: original  1P 16M 5A : 23
*** STATS: optimized 0P 9M 5A : 14
```

```
Symbols x,y,z;
ExtraSymbols vector,v;
Local F = 6*y*z^2 + 3*y^3 - 3*x*z^2 + 6*x*y*z - 3*x^2*z
          + 6*x^2*y;
Format Fortran;
Format O3,stats=on;
Print;
.end
```

- The input is a polynomial of 3 variables x, y and z.
- w/ O3, # of ops.: 23 → 12

```
      v(1)=x + z
      v(2)=2*y
      v(3)=v(2) - x
      v(1)=z*v(3)*v(1)
      v(3)=y**3
      v(2)=x**2*v(2)
      v(1)=v(1) + v(3) + v(2)
      F=3*v(1)
*** STATS: original  1P 16M 5A : 23
*** STATS: optimized 1P 6M 4A : 12
```

- Even for such a small expression, changing the optimization level can make a difference.

Horner Scheme

- For polynomials in a single variable, it is known that the Horner scheme is good:

$$\begin{aligned} a(x) &= \sum_{i=0}^{n} a_i x^i \\ &= a_0 + x(a_1 + x(a_2 + x(\cdots + x \cdot a_n))) \end{aligned}$$

- This reduces the number of operations from $\underbrace{(n-1)+n}_{*} + \underbrace{(n-1)}_{+}$ to $\underbrace{n}_{*} + \underbrace{(n-1)}_{+}$.

Horner Scheme with Many Variables

- For multivariate polynomials, one can repeatedly apply a Horner scheme in one of the variables.
- The efficiency may depend on the order in which the variables are chosen.

$$a(x,y,z) \quad = \quad y - 3x + 5xz + 2x^2yz - 3x^2y^2z + 5x^2y^2z^2 \qquad \texttt{18M 5A}$$

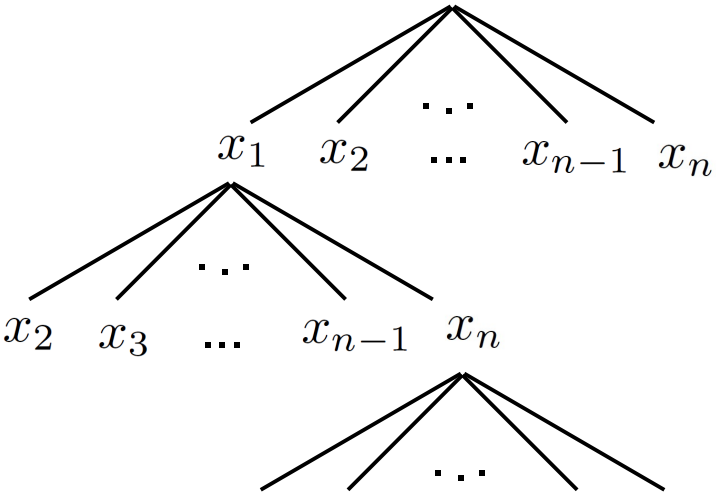
Horner Order	Formula	Operations
x, y, z	$y + x(-3 + 5z + x(y(2z + y(z(-3 + 5z)))))$	8M 5A
x, z, y	$y + x(-3 + 5z + x(z(y(2 - 3y) + z(5y^2))))$	9M 5A
y, x, z	$x(-3 + 5z) + y(1 + x^2(2z) + y(x^2(z(-3 + 5z))))$	11M 5A
y, z, x	$-3x + z(5x) + y(1 + z(2x^2) + y(z(-3x^2 + z(5x^2))))$	14M 5A
z, x, y	$y - 3x + z(x(5 + x(y(2 - 3y))) + z(x^2(5y^2)))$	11M 5A
z, y, x	$-3x + y + z(5x + y(2x^2 + y(-3x^2)) + z(y^2(5x^2)))$	14M 5A

How to choose Horner variables?

- Choose Horner variables in the occurrence order such that at every step one can get the largest decrease in the number of operations.
- Horner scheme is used with other optimization methods, e.g., common subexpression elimination (CSE). The reverse occurrence order could reduce the number of operations more in total, because more common subexpressions could appear in multiple places.
- Both may not be optimal. Try to all orders. Once the number of variables becomes large (e.g., $20! \approx 2.4 \times 10^{18}$), this is not practical.

Monte Carlo Tree Search

- The various orderings of n variables define a search tree ($n!$ paths).
- Searching through trees to find a good/best solution is common in game theory.



- Monte Carlo Tree Search (MCTS): under an assumption that good solutions are clustered in branches (seems to be true for Go game and choosing Horner variables), try many paths randomly but with taking more samples for neighborhood of good solutions.
- Monte Calro algorithm! May give a different result at each run.

Optimization Levels

- O0: Do nothing.
- O1 : Try both occurrence and reverse occurrence orderings, followed by CSE.
- O2 : Besides O1, an extra “greedy” optimization to find more common subexpressions at the end.
- O3 : The ordering is determined by MCTS with CSE and the greedy optimization.

Code Size vs. Time

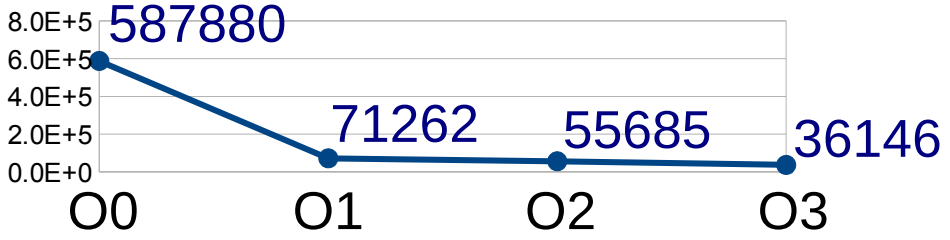
- Benchmark: The resultant of two polynomials, so-called Sylvester determinants with a number of parameters. The parameters come from two polynomials:

$$\sum_{i=0}^{n} a_i x^i = 0 \qquad \sum_{j=0}^{m} b_j x^j = 0$$

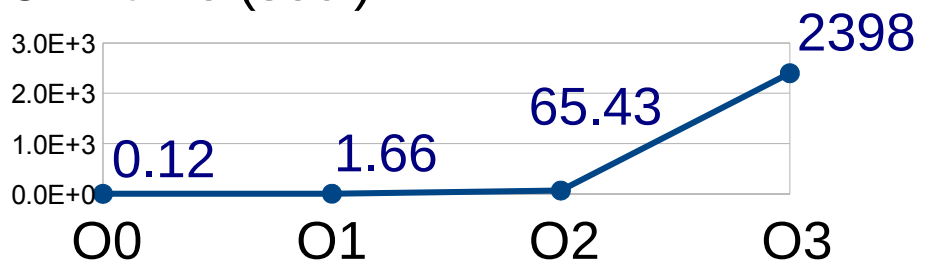
  The determinant is of a (n + m) x (n + m) matrix.

- For n = 7, m = 6 case, 13 x 13 determinant with 43166 terms.

# of operations



FORM time (sec.)

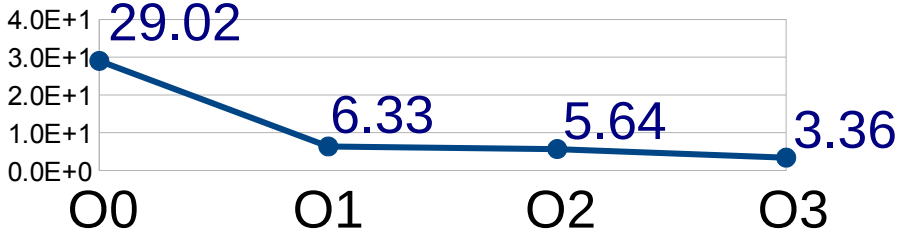


* On an Opteron 2.6 GHz processor.

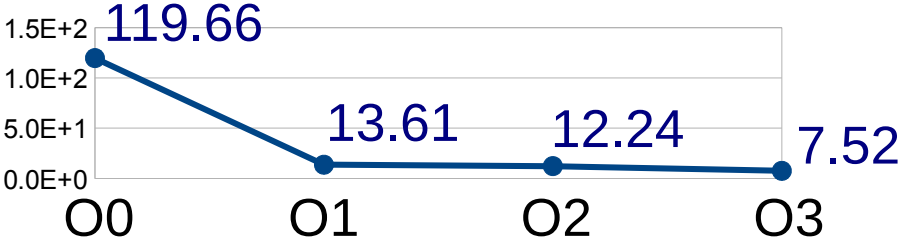
Interplay with Compilers

* With gcc 4.6.2.

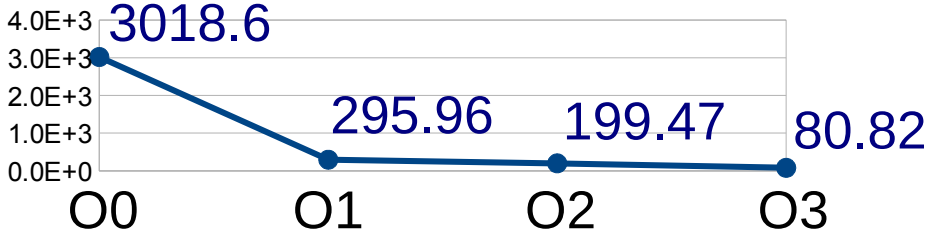
gcc -O0 compile time (sec.)



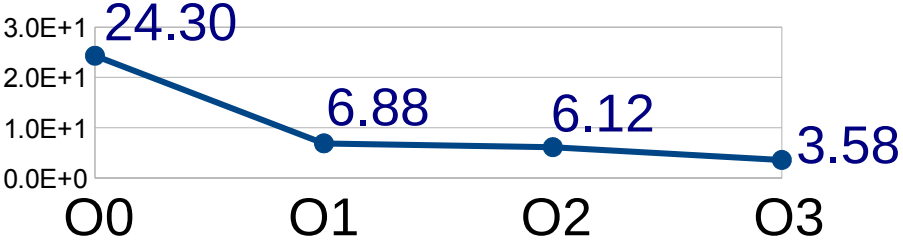
gcc -O0 run time for 10^5 evals. (sec.)



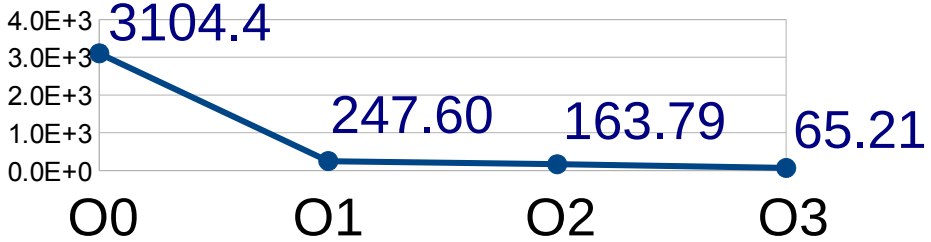
gcc -O1 compile time (sec.)



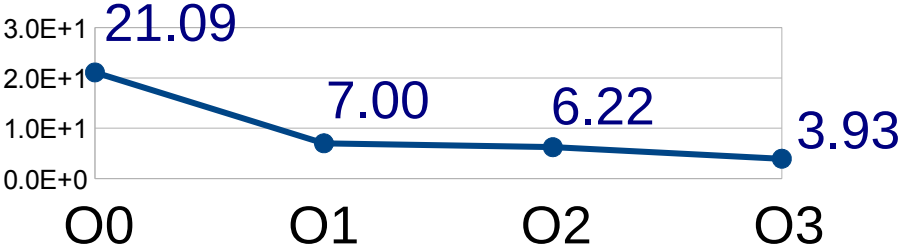
gcc -O1 run time for 10^5 evals. (sec.)



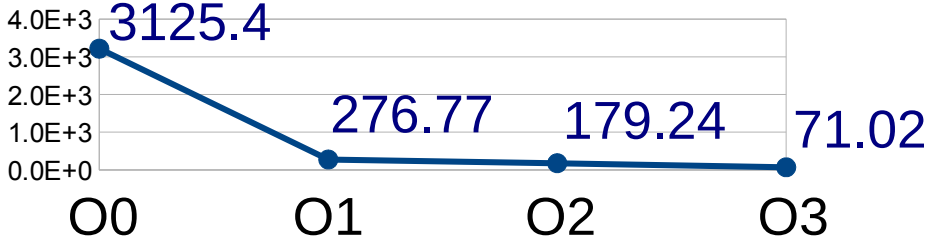
gcc -O2 compile time (sec.)



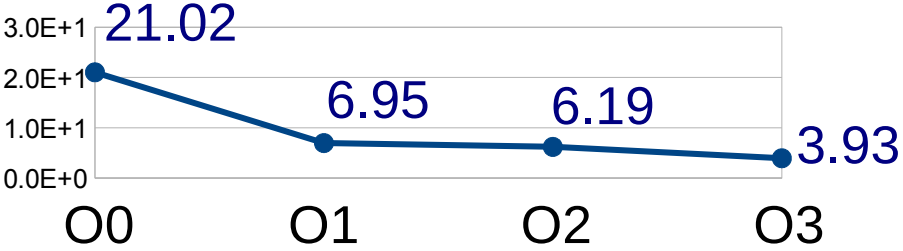
gcc -O2 run time for 10^5 evals. (sec.)



gcc -O3 compile time (sec.)



gcc -O3 run time for 10^5 evals. (sec.)



Interplay with Compilers (cont'd)

- Note: compilers cannot possibly be as efficient as FORM in their optimizations. For compilers addition is not associative because of potential numerical problems, which is ignored in FORM.
- Considering (FORM time) + (compilation time) + (run time), which combination of optimizations gives an optimal result depends clearly on the number of function evaluations needed.

Simultaneous Optimizations

- Two or more expressions can have common expressions. Combine the expressions into one expression and use Bracket statement.

```
Symbol u;
Local F, G;
Local H = u * F + u^2 * G;
Bracket u;
```

- If the expression is bracketed in terms of u, FORM does not perform the optimizations with respect to u. The optimized expression in H still contains u. One can separate the individual expressions.

Simultaneous Optimizations (cont'd)

- Example: expressions in the GRACE system, Feynman diagrams appearing in HEP. Each coefficient of the monomial with respect to Feynman parameters must be simplified. Common subexpressions appear in the coefficients.
- "# of variables m+n" means m Feynman parameters and n other parameters and the simultaneous optimization is used.

	HEP(σ)	HEP(σ)	F_{13}	F_{24}
# of variables	15	4 + 11	5 + 24	5 + 31
# of expressions	1	35	56	56
# of terms	5717	5717	105114	836010
# of ops. O0	47424	33798	812645	5753030
O1	6099	5615	71989	391663
O2	4979	4599	46483	233445
O3	3423	3380	41666	195691

Comments on Domain Specific Knowledge

- In some cases (like in expressions appearing in physics) one can do some work before using the FORM optimizations, because one has knowledge about the problem that can be useful for simplification (domain specific knowledge) while for FORM the formulas present some kind of chaos.
- The formula F_{24} can be reduced by another factor of four with a very big improvement in the speed of the optimization.

Comparison with Other Algorithms

of operations

	7-4 resultant	7-5 resultant	7-6 resultant	HEP(σ)
Original	29163	142711	587880	47424
FORM O1	4968	20210	71262	6099
FORM O2	3969	16398	55685	4979
FORM O3 [1]	3015	11171	36146	3524
Maple	8607	36464	-	17889
Maple tryhard	6451	O(27000)	-	5836
Mathematica	19093	94287	-	38102
Hypergraph + CSE	4905	19148	65770	-
Haggies [2]	7540	29125	-	13214

[1] FORM O3 run used C_p = 0.07 and 10 x 400 tree expansions.

[2] Haggies by Thomas Reiter .

Conclusion

- We have made an implementation of various simplification techniques in FORM to make the evaluation of output expressions more economical.
- The results are better than anything we could find in the literature.
- This feature will become available in the next version 4.1.