Recent developments of the FORM computer algebra system

Loop Summit 2, Cadenabbia

Josh Davies

UNIVERSITY OF LIVERPOOL

21st July, 2025

A brief history

0.0 (1984): work starts, **1.0** (1989): free, **2.0** (1991): commercial, **3.0** (2000): free

3.1+ (2006): early **`TFORM`**, **`ParFORM`** developments, **`gzip`** compression of sort files [doi]

3.2 (2007): **`#system`**, **`#pipe`**, **`#external`** [doi]
demonstrates use of e.g. **`Fermat`**, **`Reduce`** for poly GCD

3.3 (2007): **`TFORM`**, pthreads-based parallelization [doi]

3.3+ (2008): use of GMP for large integer operations, experimental **`polyratfun`** [doi]

3.3+ (2010): open source release, forum, test suite, effort to generate community involvement [doi]

4.0 (2013): improved rational polynomials, factorization, **`mul_`** etc, **`extrasymbol`**, [doi]
`transform`, checkpointing

4.1 (2013): expression optimization [doi]

4.2 (2017): **`id,all`**, improved expression optimization, polyratfun expansion, dictionaries, [doi]
spectators, 0-dim tables, **`argtoextrasymbol`**, new transform and combinatorics functions

4.2.1 (2020): topology generator, **`topologies_`** [doi]

What's next? Towards `FORM 5`

Changes over the last few years:

- Deprecations
- Bug fixes/changes
- Smaller new features
- Diagram generator
- Floating-point coefficients

Testing

Ideas for the future

- Easier, shorter term (for `FORM 5`?)
- Harder, longer term

New repository location: `https://github.com/form-dev/form`

- old link forwards to new (`https://github.com/vermaseren/form`)

Deprecations

Features which will be present in **`FORM 5`** release, but with “deprecated” status.

To our knowledge these are not used, and are a maintenance burden – maybe removed completely?

- Native Windows support: [#623]
  - Windows Subsystem for Linux (WSL) exists
- 32-bit system support: [#624]
  - various tests already fail for 32-bit builds and are skipped
  - “real physics problems” are all run on 64-bit machines
- **`ParFORM`**: [#625]
  - various tests already fail for **`ParFORM`** and are skipped
  - test suite under **`valgrind`** already disabled for **`ParFORM`** (slow)
  - **`TFORM`** scales better, and modern CPUs already out-scale **`TFORM`**
- **`Checkpoint`** mechanism: [#626]
  - current state is almost certainly buggy, not well tested

Use of these features in **`FORM 5`** prints a warning:

- Silence with **`FORM_IGNORE_DEPRECATION=1`** env. var. or **`-ignore-deprecation`** cmd opt.
- **If you use any of these features regularly, comment on the corresponding issue!**

Bug fixes/changes

Many (>50?) bug fixes made over the last few years, including:

- sorting related [#513] [#527] [#529] [#565] [#593] [#691]
- **`Load`**-ing save files [#594]
- pattern matching [#583] [#601]
- ...

Notable changes:

- Expression optimizer no longer requires output to fit in **`workspace`** [#535]
    - extra memory allocated if necessary, no need to set huge **`workspace`**
- **`multirun`** mode always used, and uses more PID digits [#591]
    - **`xformxxx.sc0`** $\longrightarrow$ **`xform1234567.sc0`**
    - **`-M`** cmd. opt. does nothing
- Fortran literal float suffix corrected [#584]
    - **`gfortran:  (Real*8):  the integer 2147483648 is too large`**
    - $\longrightarrow$ integers $\geq 2^{31}$ have a **`.D0`** suffix

Fixes/changes are all in the `master` branch — use and test this please!

Smaller new features (I)

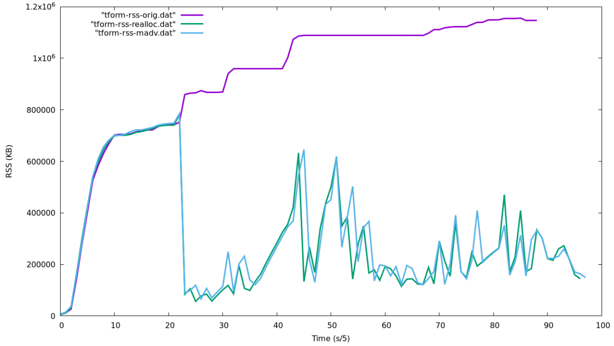
Sort buffer reallocation: (request: Markus Loechner, Zurich Workshop) [#537] [#529]

- Reallocate **`LargeBuffer`** and **`SmallBuffer`** – reduce Resident Set Size
- **`#sortreallocate`** – now, before starting this module
- **`On sortreallocate;`** – at the start of every module

✓ Useful when running with memory constraints

✘ Potentially noticeable performance impact (**`On`**: 10%? "it depends"?)

Small **`MINCER`** test:



Smaller new features (II)

Zstandard compression support: (idea: Vitaly Magerya, Zurich Workshop) [#541]

- Uses `zlibWrapper`, very little code modification [zlibWrapper]
- `On Compress,zstd;` – new default behaviour
- `On Compress,gzip;` – old default behaviour, uses `zlib`
- Simple (best case) benchmark: 8% faster, 6% smaller sort file
  - additional benefit if sort files are on slow HDD?

Read-only TableBases: (by: Florian Herren, Zurich Workshop) [#531]

- `TableBase "name.tbl" open, readonly;`
- Can now open files without write permissions
  - provide read-only `TableBase` access to collaborators
  - protect large, expensive `TableBase` from yourself!

Numerical evaluation of constants: (by: Florian Lorkowski, Zurich Workshop) [#532]

- Arbitrary-precision evaluation of e (`ee_`), γ_E (`em_`), π (`pi_`) using MPFR library

Smaller new features (III)

Backtracing: [#526]

- Effort to ease debugging, particularly for crashes of long-running jobs.
- **`On backtrace;`** – on by default, if enabled at compile time
  - use **`eu-addr2line`** or **`addr2line`** to print stack on crash (**`elfutils`**)
- Small performance impact, $\sim$1%
  - **`-g -fno-omit-frame-pointer`**, **`-rdynamic`**, form binary 2.5MB $\rightarrow$ 13MB
  - Not enabled by default, needs: **`configure --enable-backtrace`**
- **My recommendation: always enable, particularly for long-running jobs!**

```
Program terminating at gcd-simple.frm Line 10 -->
Terminate called from polywrap.cc:156 (poly_gcd)
Backtrace:
# 0: TerminateImpl at startup.c:1870:10
# 1: poly_gcd at polywrap.cc:158:32
# 2: GCDfunction3 at ratio.c:1205:2
# 3: GCDfunction at ratio.c:1061:6
# 4: Generator at proces.c:4012:9
# 5: CatchDollar at dollar.c:112:6
# 6: PreProcessor at pre.c:1129:26
# 7: main at startup.c:1746:2
```

Smaller new features (IV)

Cancel `IntoHide` plans: `NIntoHide` [#671]

- (now that **`IntoHide`** is fixed – marks all active expr for hide at module end)
- similar to **`Drop/NDrop`**, **`Hide/NHide`** etc.

Human-readable statistics: [#678]

- **`On HumanStats;`**, off by default

```
    Time =       0.00 sec    Generated terms = 1234567890 (  1 B  )
                test         Terms in output =       1234 (  1 K  )
                             Bytes used     =123456789000 (115 GiB)
```

Smaller new features (V)

FLINT interface v1: [#644]

- Interface to **`Fast Library for Number Theory`** [FLINT]
- Implements most (so far) of the **`poly`** class functionality
  - **`PolyRatFun`**, **`FactArg`**, **`FactDollar`**, **`div_`**, **`rem_`**, **`mul_`**, **`gcd_`**, **`inverse_`**
  - still missing: Expression factorization (**`Factorize`**), **`Modulus`** mode: **to do!**

- **`On flint;`** (default)
- Great performance, esp. for multivariate:
  - **`forcer`** test reduction, **`ep`**-exact
    - **753s $\rightarrow$ 521s (1.5x)**
  - **`mbox1l`** (1-loop box, vars: d, q_{12}, q_{13}, q_{33}, m^2)
    - **`mbox1l(2,2,2,1)`**: **3.0s $\rightarrow$ 1.2s (2.5x)**
    - **`mbox1l(3,2,2,2)`**: **54s $\rightarrow$ 4.0s (14x)**
    - **`mbox1l(3,3,2,2)`**: **221s $\rightarrow$ 7.7s (29x)**
  - [Takahiro's polybench]
- ~~Developed and tested with FLINT >= v3.0.1~~
  - testing since Liverpool Workshop: req. v3.2.0

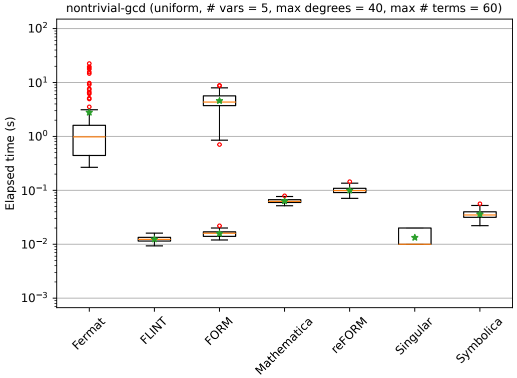


Diagram Generator

Interface to the **GRACE** generator of Toshiaki Kaneko [Comput. Phys. Commun. 92 (1995) 127-152]

- re-programmed as a **C++** library

FORM-style syntax to use it: [Manual]

- Define a **Model** containing **Particle** and **Vertex**
- **Particle particlename[,antiparticlename][,<sign><number>][,external];**
- **Vertex particle1,...,particlen:coupling;**

```
Model PHI3;
  Particle phi,1;
  Vertex   phi,phi,phi:g;
EndModel;
```

```
Model QCD;
  Particle qua,QUA,-2;
  Particle gho,GHO,-1;
  Particle glu,+3;
  Vertex   qua,QUA,glu:g;
  Vertex   gho,GHO,glu:g;
  Vertex   glu,glu,glu:g;
  Vertex   glu,glu,glu,glu:g^2;
EndModel;
```

Diagram Generator (II)

Generate diagrams using

```
diagrams_(model,set_of_input_particles,set_of_output_particles,
  set_of_external_momenta,set_of_internal_momenta,
  number_of_loops_or_coupling_constants,options)
```

For e.g.:

```
Vector Q1,...,Q7,p0,...,p21;
Set QQ:Q1,...,Q7;
Set pp:p1,...,p21;
Set empty:;
Local test = diagrams_(QCD, {glu,glu}, empty,
  QQ, pp,
  2, `OnePI_'+`NoTadpoles_'+`Symmetrize_');
```

```
test =
  - topo_(1)*node_(1,1,glu(-Q1))*node_(2,1,glu(-Q2))*
      node_(3,g,qua(-p2),QUA(-p1),glu(Q1))*
      node_(4,g,qua(p1),QUA(p2),glu(Q2))
  + ...
```

Diagram Generator (III)

Output options:

- `nonodes_`, `withedges_`, `topologiesonly_`

Filtering options: work-in-progress

- plan: align the keywords with `Qgraf` for easy transition
  - `onepi_/onepr_`
  - `onshell_/offshell_`
  - `nosigma_/sigma_`
  - `nosnail_/snail_`
  - `notadpole_/tadpole_`
  - `simple_/notsimple_`
  - `bipart_/nonbipart_`
  - `cycli_/cyclr_`
  - `floop_`

Systematic and detailed testing still required before `FORM 5` release!

Floating-point coefficients

FORM 5 has support for arbitrary precision floating-point coefficients.

- Enable with `#startfloat precision([b]its,[d]igits),MZV=weight`
  - disable with `#endfloat`.
  - coefficient printed as `float_(prec, nlimbs, exponent, limb-data)`
- `ToFloat` evaluates rational coefficients in floating-point
- `ToRational` attempts to reconstruct rational coefficients from floating-point
- `Evaluate` triggers numerical evaluation of
  - `ee_`, `em_`, `pi_`
  - `mzv_`, `mzvhalf_`, `euler_`
  - `sqrt_`, `ln_`, `li2_`, `gamma_`, `agm_`, `sin_`, `cos_`, `tan_`, `asin_`, `acos_`, `atan_`, `sinh_`, `cosh_`, `tanh_`, `asinh_`, `acosh_`, `atanh_`, (`atan2_`)
  - `lin_`, `hpl_`, `mpl_` : **work-in-progress** (**Coenraad Marinissen**)
    - Currently, uses `ginac`. Implement natively?
    - Notation? `hpl_(i1,...,in,x)`,`mpl_(lst_(i1,...,in),lst_(x1,...,xn))`

Testing

It is very helpful if people can already use the master branch for real work.

- It is supposed to be a “working version”. Nonetheless, we find a few bugs this way...
- Better to find bugs before v5 release, rather than after!

`FORM` has a test suite in the `check` directory (Jens Vollinga, Takahiro Ueda).

- Includes examples from the manual, new features, scripts reproducing (fixed) bugs.
- Runs on `GitHub`’s CI runners on commit: `Ubuntu`, `macOS`, `Windows`
  - `form`, `tform` under `valgrind`, + coverage statistics.

The tests should be (much!) more comprehensive! Makes development easier.

- Add you own tests! See `check/user.frm`.
  - Add fold containing your code `*--#[ GitHub_username_Test_name :`, and some assertions.
  - Particularly scripts with tricky performance optimizations, or use rarely-used features.
  - ☞ Should be fast-running, a few seconds at most. 30s under `valgrind`.
- Package authors should add tests! See `check/extra` directory.
  - Ensure your package is not broken by future `FORM` modifications.
- **Ask me for help!**

Ideas for the future: easier, for v5?

Various bug fixes.

`ModuleOption statistics;`

- Enable statistics printing for single module only.

`On InParallel;`

- Multi-module `InParallel;` (which is hard to use)

`Format C,_kind;`

- User-defined kind label for C print mode: C++11 has [user-defined literals].

`#printmeminfo`. Print memory usage info in log? Currently I use:

- `#pipe echo "#message Current RSS: $(($(ps -o rss= ‘PID_’)/1024))M"`

Your input here!

Ideas for the future: harder

Parallelize `Local G = F;`: loading from save files and spectators.

- This can be a big performance bottleneck for large expressions.

Compress the scratch files (`.sc0, .sc1, .sc2`) (`zlib`, `zstd`).

- Complication due to Bracket index.
  - Disable compression if an index is created?
  - Compress each bracket's content separately? (possibly poor performance)

`MAXSUBEXPRESSIONS`: remove/improve limitation?

- annoying when loading enormous text files.

Factorized `PolyRatFun`

- Factorizing denominators has been beneficial for IBP reduction (`FIRE+Symbolica`).
- Saves on `MaxTermSize` budget.

Rational reconstruction from samples over prime fields.

- `#startreconstruct ep,s,t` ?

Ideas for the future: harder (II)

Namespaces:

- currently, package/procedure variables easily clash with user scripts
- namespacing would make writing these much cleaner
- **`#namespace`**? **`#package`**?

Trace performance: more control over trace operation

- automatic replacement of scalar products generated during tracing
- cancellations: repeatedly reduce longest γ strings and sort

Improved sorting:

- Try to sort faster: make fewer comparisons
- I've had several meetings with a Liverpool CS researcher on this.
- Work-in-progress "**`powersort`**" implementation: promising.

Your input here!

Conclusions

FORM is still widely used, and will continue to be!

- used directly for computation, by many people
- used by a variety of packages
- new packages are still being developed which use **FORM**

The workshops are driving participation in development from the wider community.

- We should continue to hold them annually! Likely next: Nikhef and CERN.

There has been a lot of development over the last few years!

Aim to release FORM 5 by the end of this year.