Towards FORM 5: Status Update

FORM Developers Workshop 2025, Liverpool

Josh Davies

UNIVERSITY OF LIVERPOOL

11th June, 2025

Introduction

Progress this year:

- Deprecations
- Bug fixes/changes
- New features

Required for `FORM 5` release: (this summer?)

- Test and fix the diagram generator
- Test and fix floating-point mode
- Thorough testing of everything else!

Exercises/Ideas:

- Easier, things we could fix/add this week
- Harder, things to think about longer term (**`FORM 5.1`**?)

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

Various fixes:

- sorting related [#513] [#527] [#529] [#565] [#593]
- **`Load`**-ing save files [#594]
- pattern matching [#583] [#601]
- misc [#289] [#532]

Notable changes:

- 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

These fixes/changes (except #591) are in 5.0 and 4.3.2. branches.

New features (I)

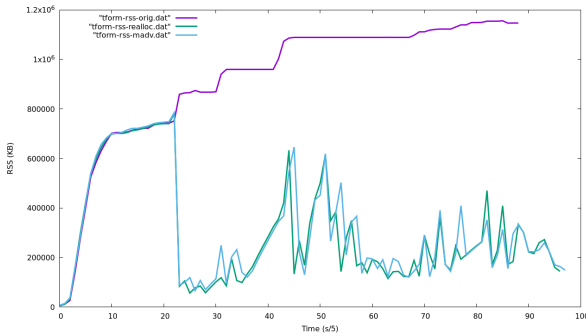
Sort buffer reallocation: (request: Markus Loechner, Zurich Workshop) [#537]

- Reallocate **`LargeBuffer`** and **`SmallBuffer`** – reduce Resident Set Size
  - possible due to split allocations of various buffers [#529]
- **`#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:



New features (II)

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

- Uses **`zlibWrapper`**, very little code modification [Facebook: 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

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, ∼1%
  - **`-g -fno-omit-frame-pointer`**, **`-rdynamic`**, form binary 2.5MB → 13MB
  - Not enabled by default, needs: **`configure --enable-backtrace`**
- **My recommendation: always enable 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
```

New features (IV)

FLINT interface v1: **This week: Fix on macOS** [#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.
- **`On flint;`** (default)
- Great performance, esp. for multivariate:
  - **`forcer`** test reduction, **`ep`**-exact
    - **753s → 521s (1.5x)**
  - **`mbox11`** (1-loop box, vars: $d, q_{12}, q_{13}, q_{33}, m^2$)
    - **`mbox11(2,2,2,1)`**: **3.0s → 1.2s (2.5x)**
    - **`mbox11(3,2,2,2)`**: **54s → 4.0s (14x)**
    - **`mbox11(3,3,2,2)`**: **221s → 7.7s (29x)**
  - [Takahiro's polybench]
- Developed and tested with FLINT >= v3.0.1
  - ✓ Ubuntu 24.04, ✘ Ubuntu 20.04
  - ☞ Debian 12, ☞ OpenSUSE Leap 15.6

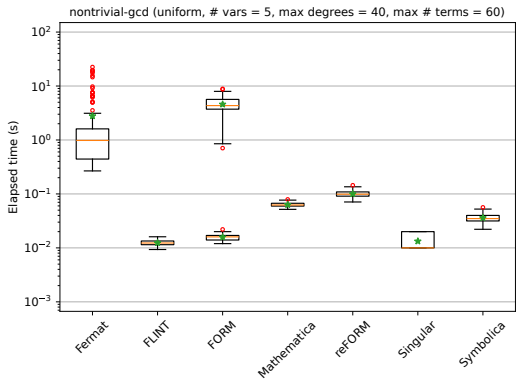


Diagram Generation

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 Generation (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 Generation (III)

There are some issues with the generator which have not been looked into closely over the last year.

E.g. when no options are given, one-loop tadpoles are not generated (but two-loop are).

Workshop Exercise: experiment with the generator!

- what works and what does not?
- what is the translation between **`qgraf`** and **`FORM`** options?
- useful info:
  - Takahiro Ueda's **`diagrammatica`**: draw topologies [Diagrammatica]
  - **`FORM`**'s interface: **`sources/diawrap.cc : GenDiagrams`**
  - **`grcc`**'s options: **`sources/grcc.cc : optDef`**
  - automatic comparison of **`FORM`** and **`qgraf`**: [Takahiro's **`qgraf.frm`**]

Floating-point mode

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

- Enable with **#startfloat precision(bits),mzv-weight**
- **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_**
  - **lin_**, **hpl_**, **mpl_** : (not yet merged) **See Coenraad's talk**
- **ToFloat** evaluates rational numbers in floating-point
- **ToRational** attempts to reconstruct rationals from floating-point

Still to do:

- **atan2_**
- add test cases for all of these functions
- problems with evaluation at low precision -> try to fix this week

Testing

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

- Includes examples from the manual, new features, scripts reproducing (fixed) bugs.
- After making changes, run tests locally with **make check**.
- Runs on **GitHub**'s CI runners on commit: **Ubuntu**, **macOS**, **Windows**
  - **form**, **tform** under **valgrind**, + coverage statistics.

The tests could be much more comprehensive!

- 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.

Contributing: (tests and code changes)

- Work on your own fork. Iterate there, **push --force**, etc.
- Create a clean pull request back into **form-dev/form**.
  - Each merge needs to "work properly" (for easy bisecting), CI helps enforce this.

Exercises/Ideas: Easier (probably?)

Fix `ranperm_(t,?a)` when `t` is a tensor.

- `ranperm_(t,1,2,3) -> t(20_,2,20_,3,20_,1)`

Fix or forbid `replace_(fun,tens)` at compiler level and runtime (from dollar var)

- `f(i1,i2,i3) -> t(?,i1,?,i2,?,i3)`

Fix [#615]: `sum` does not trigger `repeat`.

- Do any other statements not set `RepPoint`?

`ModuleOption statistics;`

- Enable statistics printing for single module only.

`On InParallel;`

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

`On humanstats;` (suggest refactoring existing stats code first `snprintf?`)

```
Time =       0.00 sec    Generated terms = 1234567890 (1.2G)
           test          Terms in output =       1234 (1.2K)
                         Bytes used      =123456789000 (115GiB)
```

Exercises/Ideas: Easier (probably?)

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

- `Format C,_kind;`

`#printmeminfo`. Currently I use:

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

Fix [#627] Particle definition can be overwritten as function. If with symbol:

- `test.frm Line 5 --> f has been declared as a function already`

Fix [#612] why doesn't `#timeout` work properly with `tform`?

- This causes Github CI jobs to fail occasionally.

Fix [#646] `Evaluate` on `mzv_`, `euler_` with no arguments crashes.

Fix [#267] Crash in `Transform f addargs(1,last);` if `f` appears with no arguments.

- Bonus: missing mutex `UNLOCK` in `RunPermute`.

Fix [#445] Tensors with 0 for argument vanish.

Exercises/Ideas: Easier (probably?)

Fix [#633] formatting of **`FactArg`** for negative vector arguments.

- **`f(-p) -> f(p,-1,1)`**

Fix [#642] **`putfirst_`** with negative arguments.

- crashes or produces nonsense

Fix [#631] builtin macros destroy **`?a`**, why? Note **`?a`** appears explicitly in the source...

```
#procedure foo(?a)
  #message 'toupper_(abc)'
  #message '?a'
#endprocedure

#call foo(1,2,3)
.end
```

```
~~~ABC
foo Line 3 ==> Undefined preprocessor variable ?a
```

Exercises/Ideas: Harder, longer term. Not for 5.0. But 5.1?

Parallelize **`Local G = F;`** and loading from spectators.

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

Compress the scratch files (**`zlib`**, **`zstd`**).

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

Remove/improve **`MAXSUBEXPRESSIONS`** limitation?

- Annoying when loading enormous text files.

`#includemma` load Mathematica-format text files directly?

Factorized **`PolyRatFun`**? **`FactPolyRatFun`**? **`PolyFactFun`**?

- Factorizing denominators has been beneficial for IBP reduction (**`FIRE+Symbolica`**).
- Saves on **`MaxTermSize`** budget.
- **`fprf(num, lst_(den1,pow1), lst_(den2,pow2), ...)`** ?

Facilities for rational reconstruction from samples over prime fields. **`#startreconstruct ...`**

Conclusions

There has been a lot of development over the last year!

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 some participation in development from the wider community.

- We should continue to hold them annually!

There is still a list of things to do, for the release of **FORM 5** (this summer?).

- **Let's try to resolve at least some this week!**